Technische Universität Graz
Erzherzog-Johann-Universität
Institut für Maschinelles Sehen
und Darstellen

Diplomarbeit aus Telematik

Machine Vision Driven Real-time Blackjack Analysis

Arno Zaworka

Graz, August 2001

Betreuer
Univ.-Ass. Dr.techn. Dipl.-Ing. Stefan Scherer

Begutachter
O.Univ.-Prof. Univ.-Doz. Dr.techn. Dipl.-Ing. Franz Leberl

VISIT…

Abstract

The potential of solving real-time demanding industrial applications, using vision-based algorithms, drastically grew due to an increasing availability of computational power.

In this thesis a novel real-time, vision-based Blackjack analysis system is presented. The embedding of the vision algorithms in a compound system of other information sources such as an electronic chip tray, reduces the vision task to detect cards and chips. Robust results are achieved by not just analyzing single frames but an image stream regarding game-flow informations. The requirements for such a system are a highly robust and adaptive behavior. This is motivated by the vital interest of casino entrepreneurs in a 100 % statistical analysis of their offered gambling in order to support the business plan, measuring table and dealer performance and give accurate player rating. Extensive experiments show the robustness and applicability of the proposed system.

Kurzfassung

Die Entwicklung leistungsfähiger Hardware erlaubt die Anwendung von komplexen Bildverarbeitungsalgorithmen in industriellen Echtzeitsystemen.

In dieser Arbeit wird ein neuartiges, echtzeitfähiges Blackjack-Spielanalysesystem mittels Bildverarbeitung präsentiert. Das Einbetten der Bildverarbeitungsalgorithmen in ein kombiniertes Gesamtsystem mit anderen Informationsquellen wie einem elektronischen Chip-Tray verringert die Bildverarbeitungsaufgaben auf die Erkennung von Karten und Chips. Robuste Resultate werden erzielt, indem man nicht nur einzelne Bilder, sondern den Bildfluss bezüglich Spielabläufen analysiert. Die Anforderungen für ein solches System sind ein in hohem Grade robustes und anpassungsfähiges Verhalten. Dies wird durch das wesentliche Interesse von Casino-Unternehmern an einer 100 % statistischen Analyse ihrer angebotenen Spiele zur Unterstützung des Unternehmenssplans und Messung von Tisch- und Kartengeberleistung motiviert und ermöglicht genaue Spielerbewertung. Umfangreiche Experimente zeigen die Robustheit und die Anwendbarkeit des vorgeschlagenen Systems.

Acknowledgments

This thesis is the result of my work from the last three years at the Institute for Computer Graphics and Vision (ICG) at the Technical University of Graz. I would like to thank my supervisor Stefan Scherer for his scientific guidance and enduring support throughout this work, but also my fellows on the ICG for helpful discussions and comments to the making of this thesis and appropriate implementations, especially the GIPLIB development team. The financial support of GRIPS Electronic GmbH is grateful acknowledged.

Finally I would like to thank my parents for their patience and support during all the years of studying.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The aim of this thesis was the development of an computer aided system, which is capable to analyze the card game Blackjack and provides various output information for statistical purposes. Because of the increased performance of computer systems and peripheral components the application of machine vision algorithms in a large variety of industrial real-time applications [BBR+99] is possible. Captured image data can be analyzed just in time, storing and post-processing in the case of huge amount of captured data is obsolete. Regarding to this trend, a vision-based approach was implemented and evaluated.

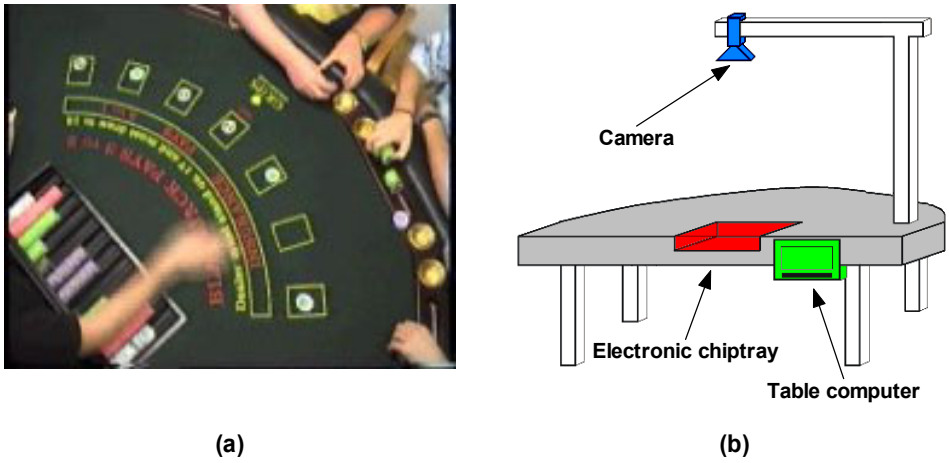


Figure 1.1: Blackjack analysis: (a) captured game scene sample; (b) schematic setup.

1.1 Motivation and Purpose

The development of analysis systems for table games like Blackjack is motivated by the need of statistical informations on the part of casino entrepreneurs. Conditional upon the huge amount of money in casino business, marketing informations and business plan

support is needed which can be extracted from achieved analysis data. The benefit of such data is widespread:

- table statistics
- table accounting
- player rating
- dealer assessment.

Measurement of table and dealer performance can be used for identification of best-in-class employees and optimization of the efficiency of the team. Financial success and working efficiency, even patron attraction can be derived. Accurate player rating can lead to player profile creation, the win/loss results per patron, amount of played games and average bet are some examples.

The requirements to an analysis system, which is capable to provide the needed informations are that it is highly adaptable to different tables, easy to install and use, but also inconspicuous to the dealer and players. The dealer should not be restrained in their usual work. Beside that it is clear that the analysis algorithms must work highly robust and the resulting output data must be very accurate.

The output of the system should contain informations like the amount of played games on a Blackjack table, amount of participating players, their Bust and Blackjack occurrences and also dealer characteristics and the chip-flow among the chip-tray and the player boxes.

1.2 Realization

In this thesis a low budget vision-based real-time analysis system of the Blackjack game is presented. A mounted table camera provides image data of the game table which is processed with an embedded table computer. Extracted information like visible game cards and chip-stacks on the table is combined with information obtained from an electronic chip-tray in order to analyze game-flow and to determine the different game states which are further described in chapter 2. Image processing is separated into the tasks robust card- and chip-detection. In order to reduce the computational costs and therefore increasing the performance, the captured images are separated into regions of interest which can be easily modified at demand for different game tables. Within these regions, the objects to be detected are classified by calculating global descriptions through a low-level image processing chain consisting of adaptive segmentation, morphological and logical operations and Hough analysis, but also using additional logical sequence controlling. Figure 1.2 shows a brief overview of the system containing image acquisition, the vision modules for object detection and the logical game-flow controller which handles the received data from the vision modules and calculates resulting output information.

The experimental setup was designed in cooperation with GRIPS Electronic GmbH [GRI99], the industrial partner in this project. The research and implementation of a

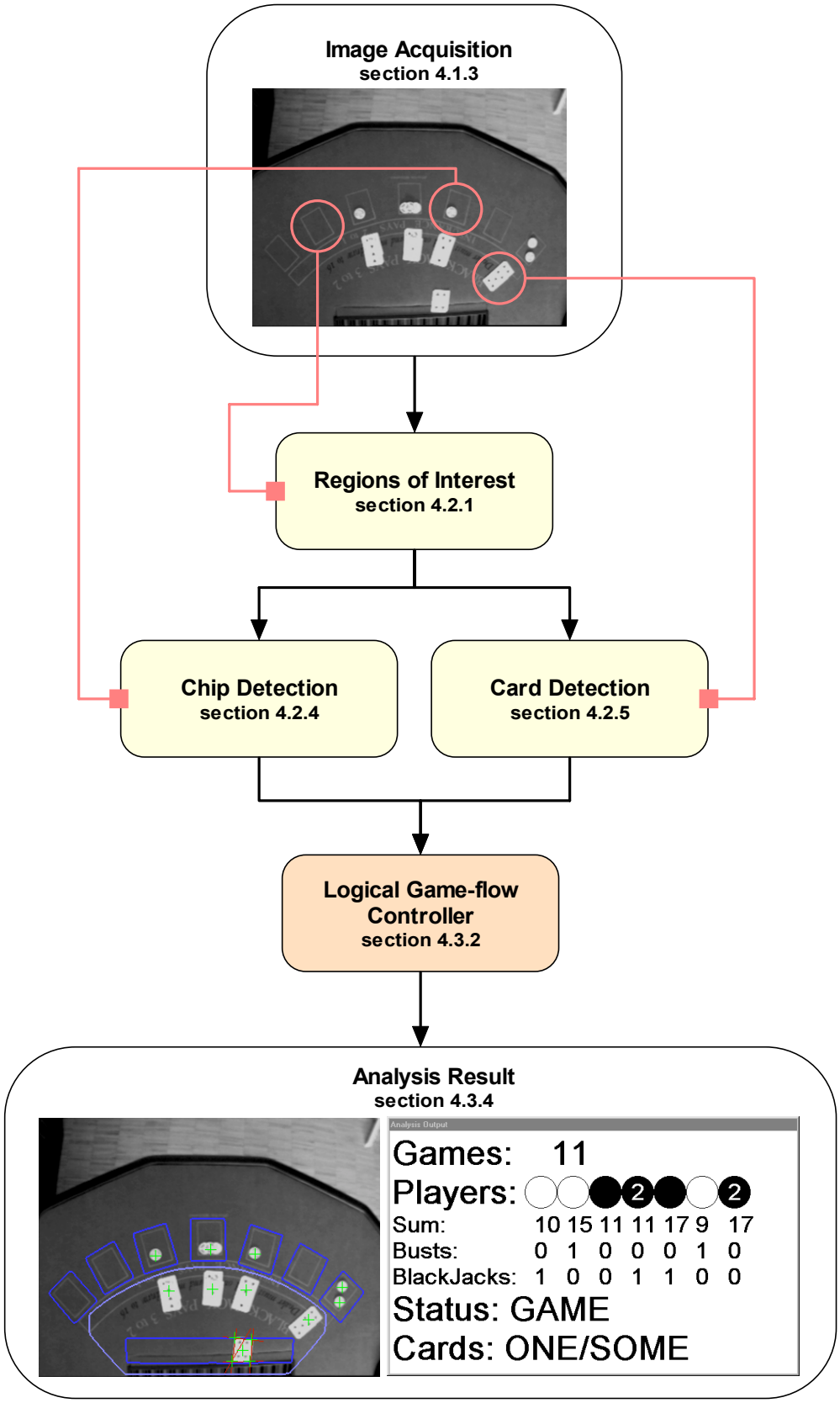


Figure 1.2: Analysis procedure overview.

prototype system is done at the Institute for Computer Graphics and Vision (ICG) [Ins01] at the Technical University of Graz in recent years. The used programming language is C++ with the use of the GIPLIB [Gip98], a general image processing library which is build on the ICG.

Initial results of the proposed system were presented at the 24th Workshop of the Austrian Association for Pattern Recognition (ÖAGM/AAPR) in Villach, Carinthia, Austria [ZS00].

Regarding the integrated game analysis system, this thesis is the first vision-based work known to the author. From the vision point of view the state of the art of the related image processing algorithms is further discussed in the subsequent chapters.

1.3 Document Structure

This thesis is divided into six chapters. The following chapter after this introduction illustrates the game Blackjack itself, describes the possible player actions and the dealer attitude. A detailed enumeration of the desired analysis output is given and the subdivision of the game into different game states is presented, which is needed for proper analysis.

Chapter 3 gives an overview of related image processing algorithms which are used and evaluated in this theses with various introductive examples. The main parts are thresholding techniques for fast segmentation, basic morphological operations and the Hough transformation. The whole developed system is presented in chapter 4. The experimental setup used for the prototype is described with its game and hardware resources. The used vision modules are detailed shown in section 4.2, which is one of the key parts of this thesis. The separation of the captured image in regions of interests, learning of chip and card features and finally the detection and classification of chips and cards are the main topics.

The whole working combination of hardware and vision algorithms with additional game-flow dependencies and cognitions is listed up in chapter 4.3. The problem specification fulfillment and possible additions to the prototype are given as well as its user interaction possibilities including the graphical user interface.

Chapter 5 shows the robustness of the chosen algorithm and presents experimental results. Stand-alone chip- and card-detection but also overall game behavior is analyzed under various conditions.

Finally conclusions and an outlook of possible enhancements is given in chapter 6.

Chapter 2

The Blackjack Game

Blackjack or "21" is the casino's most popular table game [Don00]. The rules are simple to learn, but vary slightly depending on the location. In this chapter a basic game description is presented, which contains the used rules and points out the needed details for this thesis. Furthermore the desired output demands on an analysis system are specified. An in-depth breakdown of the game leads to the method how this requirements can be met.

2.1 Game Description

Blackjack is played with one or more decks of 52 cards. Cards are valued at face value, except for the aces, which can be valued at either one or 11 points. Face cards - jacks, queens and kings - are valued at 10 points. The object of the game is to beat the dealer. A player does this by having a higher valued hand than the dealer, while not "busting" by exceeding 21 points. If either the player or dealer busts, the other wins automatically. A major advantage for the dealer is that the player must play his hand first. Even if a player busts and the dealer subsequently busts as well, the player loses. Ties are a standoff and neither wins. If either the player or dealer is dealt an ace and 10-valued card, he has a Blackjack. A Blackjack is an immediate winner and pays the player at 3-2 odds. Other winning hands are paid at 1-1 or even money. The play begins with the dealer distributing two cards to each player and two to himself, dealt one at a time with the first card going to the player. One of the dealer's cards is exposed to help the player decide whether to hit (take another card) or stand (play the ones he has). In addition to standing pat, the player may double down his hand or split pairs. By doubling down, the player doubles his initial bet and accepts one additional card to complete his hand. By splitting pairs, the player also doubles his bet, but separates his two identical cards into two separate hands, to which he draws additional cards. Each hand is played separately, taking hits as needed. Once all the players have taken their turn, the dealer takes his second card and turns it face up. If his total is 17 or greater, the dealer stands, else he must take a card until the total reaches 17 or greater.

A typically Blackjack playground in a casino consists of a usually half-circled or kidney shaped table with the dealer standing midway at the flat part where the chip tray is located. On the other side 6 or 7 high-back, stool-type chairs are located for each player. The conventional game played in most casinos today is dealt from a heavy plastic rectangular box called a "shoe", but some offering also hand-dealt games. Shuffling the cards varies, most casinos have a particular shuffle method the dealers are instructed to use to insure that the cards are in a random sequence. Some casinos employ a mechanical shuffler to cut down the amount of wasted time.

2.2 Desired Output - Problem Definition

Game analysis is a widespread term, therefore a clear definition of the behavior and tasks for the desired Blackjack analysis system is needed. The main part of the problem definition is the desired output information which should be provided by the system. In this work the major goal is the ascertainability of various outputs for further statistical calculations. Those game informations must be very exact and the extracting algorithms must work robust regarding to occurring light environment changes and table vibrations. Another criterion for optimization is the analysis runtime, real-time capabilities must be achieved. The requirements on the quality of each task is different, thus three rating levels are defined:

- 1 = mandatory
- 2 = important
- 3 = nice.

The detection of the following topics is the major task of the analysis system, the number in the brackets is the rating level of the specific task:

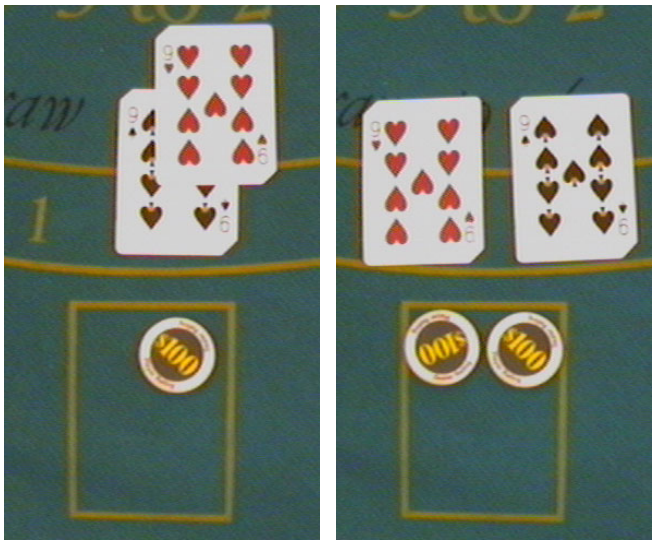
- Player Actions
  - Bust (1) - The players hand is exceeding 21 points and his chip pool is immediately taken to the chip tray by the dealer, it is not allowed that the dealer keep some chips in his hand for later payouts.
    Needed information: the position (player box) from which the chips were removed.
  - Blackjack (2) - True 21 points are reached with an Ace and another 10 point valued card. This is an instant winner and the player gets three half of his bet paid out of the chip tray. The new chips are put beside and on the initial stack as shown in 2.1. They are never placed on one single pile. In some casinos the win is not paid immediately and thus the separate detection is not needed.
    Needed information: the position (player box) in which the chips are placed.

(a) (b)

Figure 2.1: Blackjack situation: (a) Initial bet; (b) Three half payout.

  - Split (3) - With two equal valued cards the player may split his hand. Therefore he doubles his initial bet and the dealer separates his two identical cards into two separate hands as shown in 2.2.
    Needed information: the position (player box) where the split occurs.
  - Double down (3) - The player also doubles his initial bet, but his hand is not split. Instead it is completed with one additional card.
    Needed information: the position (player box) where the double down occurs.
  - Tip bet (3) - The player puts down an additional bet outside of his usual playing box that indicates a tip bet for the dealer. If the player wins the additional chips are the dealers personal profit.
    Needed information: the position (player box) where the tip bet occurs.

- Amount of played games (1) - Permanent counting of played games to determine the throughput.
- Participating players (2) - A player joins the game by putting down a bet in his box. Usually one player uses one box, the bet is one single stack, but he may use more boxes if they are free. Furthermore he can also bet on other players by putting down chips in their boxes. Those chips are on a separated stack, its not allowed to put them on the owners chip stack. As output the amount of used player boxes and the number of chip piles within is needed, not the amount of single chips nor their value.
- Shuffling (3)- If all cards up to a solid-color plastic card called the "cut-card" are used in the shoe, re-shuffling will be done after the current game. This is done manually



(a) (b)

Figure 2.2: Splitting: (a) Initial bet, two equal valued cards; (b) Splitted hand, additional independent bet.

by the dealer on the game table or mechanically nearby. The cut-card is afterwards inserted into the shoe by a player indicating the next shuffling break.

Beside those tasks the system must be aware of the detection of different game states for proper analysis. Those states and their relations are described in the next section.

The problem set fulfillment of the implemented prototype is shown in section 4.3.3, it meets all of the mandatory requirements and can be used also for additional information output.

2.3 The Blackjack Flowchart

The Blackjack game is very simple, nevertheless it can be separated into four different game states:

- Idle
- Game
- Payment
- Shuffle.

The differentiation into states is necessary to distinguish the specific meaning of dealer and player actions without knowing the overall game tenor. Prerequisite for setting up an

analysis system is the knowledge about the relation between these game states. Figure 2.3 outlines the relations in the game flowchart.

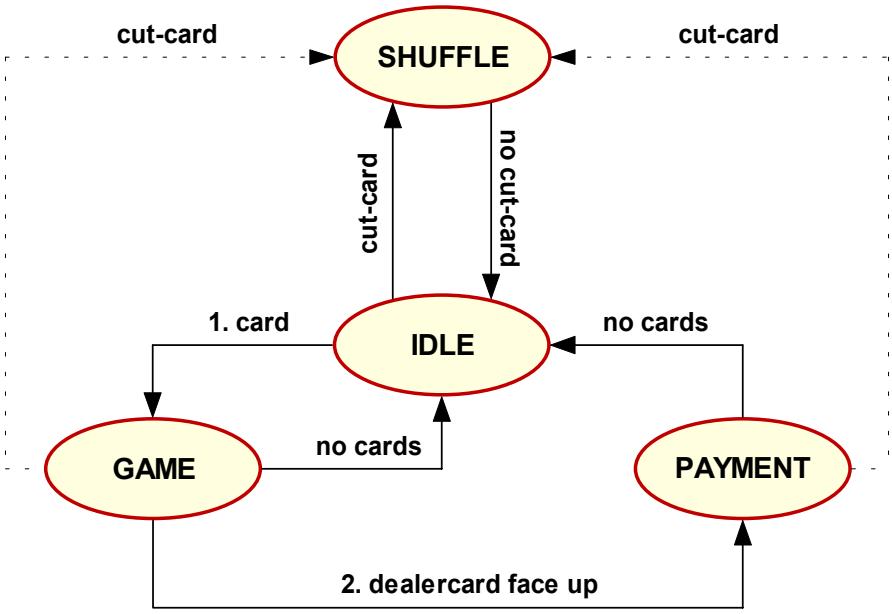


Figure 2.3: Blackjack Flowchart.

If there are no cards on the table the *Idle* state is set. At this time players are allowed to move chips, set their pool in the players boxes and give a tip bet. After the players are done the dealer starts to hand out cards beginning with the left-most player. When the first card is put on the table the *Game* state is reached. From now on no more chip movements or bets are allowed by the players until they had lost or won in the current game and bet for the next. The only exception is split or double down where the player must double his initial bet. The dealer satisfies each player with cards, player Busts and Blackjacks may occur and are immediately handled. At least he starts giving himself additional cards which initiates the *Payment* state. Players obtain or loose chips depending on their cards and the dealers cards. This part is skipped if all players went busted. On clearing the table from all cards the *Idle* state is reached again and one game is over.

After several games the appearing cut-card in the dealing shoe indicates reshuffling. The cut-card is put down the table initiating the *Shuffle* state after the current game is played to its end. To point out that the *Shuffle* state is not immediately set after the trigger occurs, the connections in Fig. 2.3 are dotted. The use of a cut-card and the task of shuffling is varying very much from casino to casino which is the reason for the low rating level for the detection of this state.

Summarizing the above, this states, actions and triggers are possible:

- Idle state - No cards are on the table

  - Bet - Put chips into the players regions
  - Tip Bet - Put chips beside the players regions
  - Trigger Game state - Put down the first player card on the table
  - Forward to Shuffle state if cut-card is on table

- Game state - Not more than one dealer card is faced up
  - Split - Separate cards and double initial bet
  - Double down - Double initial bet
  - Bust - Exceed 21 points
  - Blackjack - Reach true 21 points
  - Trigger Payment state - Put down or face up second dealer card
  - Trigger Idle state - Remove all cards from table
  - Mark next Idle state for Shuffle state forward - Put down cut-card

- Payment state - More than one dealer cards are faced up
  - Trigger Idle state - Remove all cards from table
  - Mark next Idle state for Shuffle state forward - Put down cut-card

- Shuffle state - Cut-card is on the table
  - Trigger Idle state - Remove cut-card from table

The distinction of the Blackjack game into the presented states and the relations between them is simplifying the whole analysis problem and the possible player and dealer actions are observable more easily. The use of the presented cognitions can be found in section 4.3.

Chapter 3

Basics of Related Algorithms

In this chapter related image processing algorithms to this thesis are described. It is an overview of the major used methods and can be used as reference for people that are not very familiar with image processing. In this chapter only the algorithms themselves are presented separately, their behavior due to this work and cooperation to each other is described in section 4.2 where they are evaluated and most are used in the final prototype.

The following sections describing morphological algorithms, which may be used for binary image manipulations, the Hough transformation for extracting simple objects out of an image, thresholding techniques for segmenting gray level images in objects and background, image convolution for smoothing or edge extraction, border tracing for object boundary extraction and region labeling for object separation.

3.1 Mathematical Morphology

Mathematical morphology is based on the algebra of non-linear operators working on object shape and stems from the set theory. It is used mainly for purposes like image preprocessing, enhancing object structure, segmenting objects from the background and object quantification where it is often better and faster than standard approaches [SHB93].

The main protagonists of mathematical morphology were Matheron and Serra [Ser82] which initiated set formalism for binary image analysis. That highly mathematical theory is omitted in this section and only some basic morphological operations for image processing are listed and further described with practical examples.

Because morphology is set theory, real images must be modeled using **point sets**. For binary images each point is represented by a pair of integers ($\in Z^2$) that give it's coordinates. For grey scale morphology sets of integer triples ($\in Z^3$) are used. Figure 3.1 shows an example for a point set of a binary image where the points belong to the object represented by the black squares. The origin is marked as a diagonal cross, points can be treated as a vector (row,column) with respect to that origin (0,0).

$$X = \{(1,1),(1,2),(1,3),(1,4),(2,4),(3,2),(3,3),(4,1),(5,1),(5,2),(5,3),(5,4)\}$$

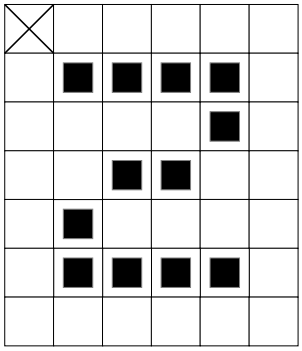


Figure 3.1: A point set sample.

A morphological transformation Ψ is given by the relation of the image (point set X) with another small point set B called **structuring element**. To perform transformation $\Psi(X)$ on the image X means that the structuring element B is moved systematically across the entire image. The result of the relation between X and B in each position forms the output set of points.

3.1.1 Binary Dilation and Erosion

Dilation and Erosion are the primary morphological operations. Most of the more complex operations like opening, closing and hit or miss transformation are based on them. The following descriptions are using the Minkowski's formalism [HS92] which is close to standard mathematical notions.

Dilation

The morphological **dilation** $\oplus$ combines two sets using Minkowski set addition $((a,b) + (c,d) = (a+c, b+d))$. The dilation $X \oplus B$ is the point set of all possible vector additions of pairs of elements from each set.

$$X \oplus B = \{p \in Z^2 : p = x + b, x \in Xb \in B\} \tag{3.1}$$

Figure 3.2 illustrates an example of dilation.

$$\begin{gathered} X = \{(1,1),(1,2),(1,3),(1,4),(2,4),(3,2),(3,3),(4,1),(5,1),(5,2),(5,3),(5,4)\} \\ B = \{(0,0),(0,1)\} \\ X \oplus B = \{(1,1),(1,2),(1,3),(1,4),(2,4),(3,2),(3,3),(4,1),(5,1),(5,2),(5,3),(5,4), \\ (1,5),(2,5),(3,4),(4,2),(5,5)\} \end{gathered}$$

Figure 3.3 shows an image and the result of the dilation using a $3x3$ square as structuring element. It points out that dilation can be used for isotropic grow of objects. Furthermore the operation is commutative, associative and invariant to translation. Common use for

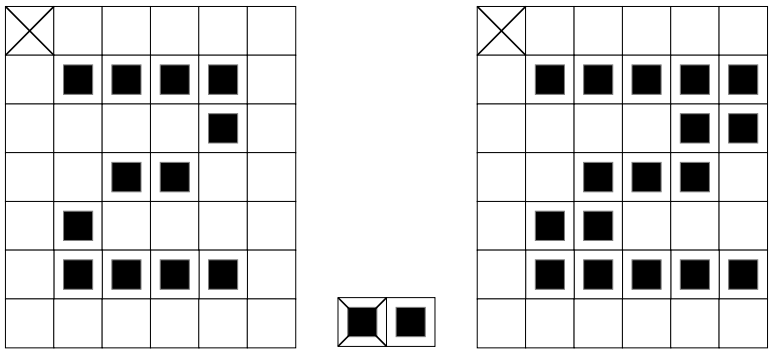


Figure 3.2: Dilation.

dilation is filling small holes and gaps in objects and smoothing of object boundaries. It enlarges the object and connects neighboring particles.

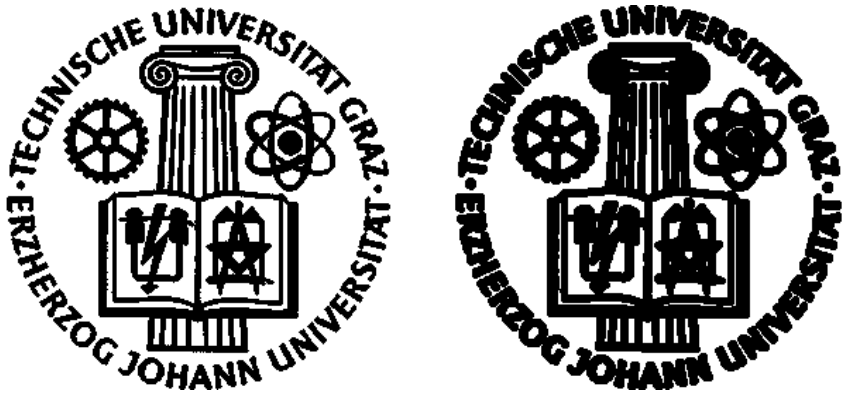


Figure 3.3: Dilation as isotropic grow.

Erosion

Erosion $\ominus$ combines two point sets using Minkowski set subtraction and is defined as

$$X \ominus B = \{p \in Z^2 : p + b \in X \forall b \in B\} \tag{3.2}$$

The result of the erosion is given by all points p for which all possible vectors $p + b$ are in X. Figure 3.4 illustrates an example.

$$X = \{(1,1),(1,2),(1,3),(1,4),(2,4),(3,2),(3,3),(4,1),(5,1),(5,2),(5,3),(5,4)\}$$
$$B = \{(0,0),(0,1)\}$$
$$X \ominus B = \{(1,1),(1,2),(1,3),(3,2),(5,1),(5,2),(5,3)\}$$

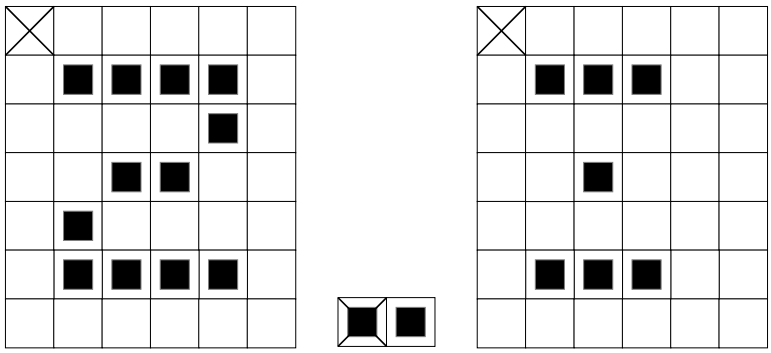


Figure 3.4: Erosion.

Optional it can explained by moving the structuring element over the desired image and check for each position if the structuring element is full within the image object. The eroded result is the set of points where that check is true [Soi99]. Erosion can be used to remove small objects and small fingers or peninsulas, it smooths object boundaries and shrinks objects. It is invariant to translation but not commutative like dilation. Figure 3.5 shows erosion as isotropic shrink used a $3x3$ square structuring element. Another ability of morphological transforms is to find object contours very quickly. This can be achieved for example by subtracting the eroded image from the original one as shown in Figure 3.6.

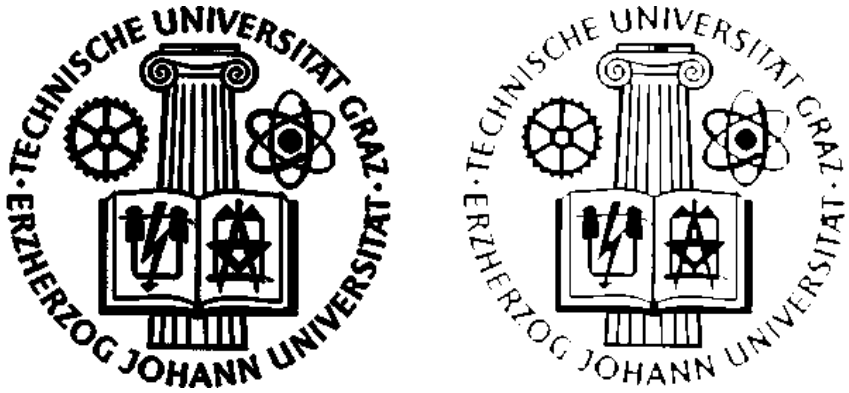


Figure 3.5: Erosion as isotropic shrink.

Erosion is the dual operator of dilation, but both are not reversible transformations. That means the result is the same if an object in an image is eroded or the background is dilated, but if an object is eroded and afterwards dilated with the same structuring element it must not have the similar shape from the original object. Instead the result may be simplified and less detailed. Combinations of erosion and dilation lead to important morphological transformations described in the following section.

Figure 3.6: Contours by subtracting the eroded image from the original one.

3.1.2 Opening and Closing

Erosion followed by dilation creates the morphological transformation called **opening**. The opening of an image X by the structuring element B is defined as

$$X \circ B = (X \ominus B) \oplus B \tag{3.3}$$

Opening can be used for similar tasks to erosion. Objects and fingers smaller than the structuring element can be removed, the difference to erosion is that the remaining objects are not getting smaller and stay in almost the original shape. Figure 3.7 shows the difference between erosion and opening from the same original image with the same structuring element. It's dual operation is called **closing** which is dilation followed by erosion.

$$X \bullet B = (X \oplus B) \ominus B \tag{3.4}$$

Closing connects objects that are close to each other, fills up small holes and smooths the object outline by filling up narrow gulfs same as dilation. But the remaining object shape is not getting bigger and stay in almost the original shape as well.

Openings and closings are idempotent, that means reapplication of these transformations does not change the previous result. Iterative opening can be done by n erosions followed by n dilations, iterative closing by n dilations followed by n erosions.

3.2 Hough Transformation

The Hough transformation is an method for finding objects within an image if the shape or the size of the desired objects is known. It is very effective because it can even be used successfully at overlapping or semi-occluded objects and is not sensible to noise. Typical

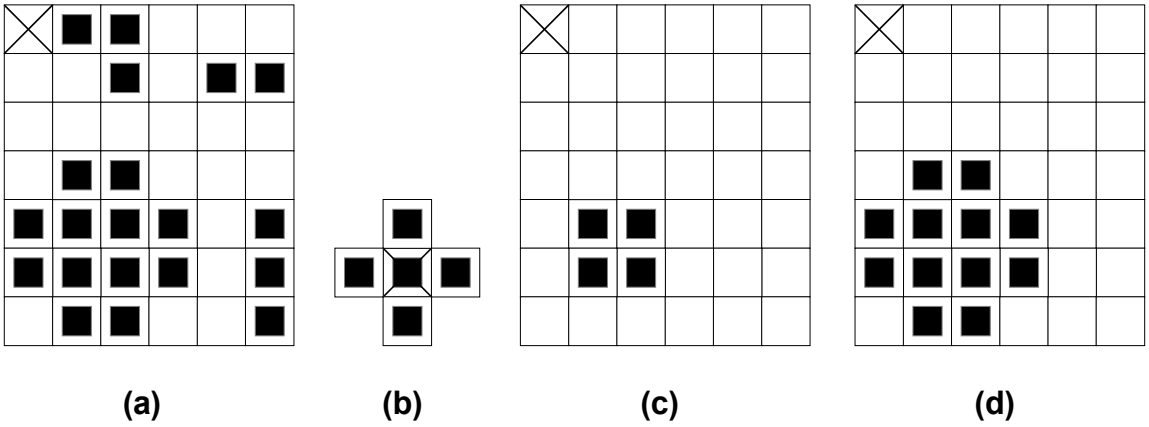


Figure 3.7: Difference between erosion (c) and opening (d) of the same original image (a) and structuring element (b).

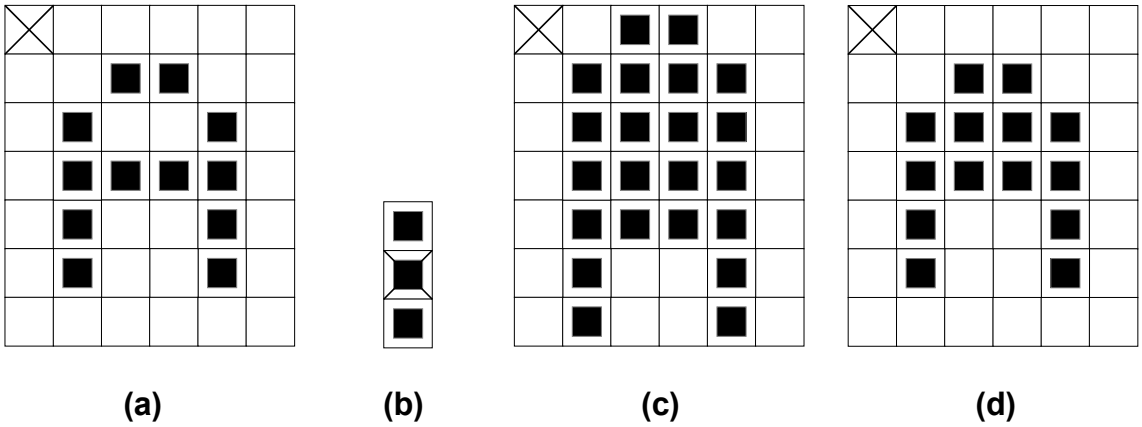


Figure 3.8: Difference between dilation (c) and closing (d) of the same original image(a) and structuring element (b).

tasks are to locate lines or circular objects, but even complex shapes can be found by generalized Hough transforms [Bal81].

3.2.1 Hough Transform principles

The original Hough transform was designed to detect straight lines and curves [Hou62]. The basic idea of this method can be illustrated by the simple problem of detecting straight lines in an image [DH72]. A straight line is defined by two points $A = (x_1, x_2)$ and $B = (x_2, y_2)$ as shown in figure 3.9 (a)). All straight lines through point A can be represented by the equation $y_1 = kx_1 + d$ for different values of k and d. Now the same equation is interpreted as an equation in the parameter space k, d; all straight lines through point A are then represented by $d = -kx_1 + y_1$ (see figure 3.9 (b)) and all straight lines through point B can represented by $d = -kx_2 + y_2$ as well. The only common point of both straight lines in the k, d parameter space is the point which represents the line connecting points A and B in the original image space. This means that any line in the original image is represented by a single point in the k, d parameter space and any part of this line is transformed into the same point. The main idea of line detection is to determine all possible line pixels in the image, to transform all lines that can go through those points to the parameter space k, d and find points in the parameter space that frequently resulted from that transform.

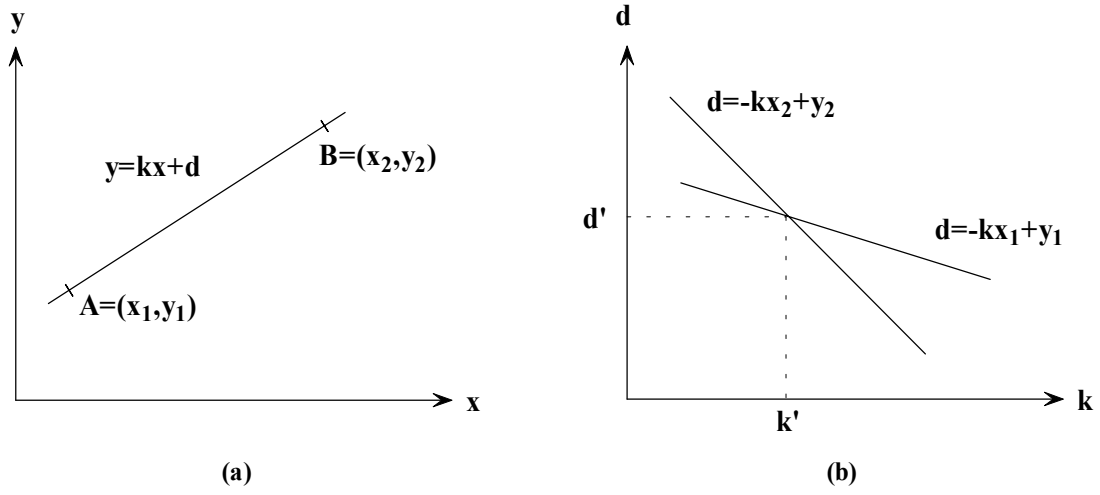


Figure 3.9: Hough transform principles: (a) image space; (b) k, d parameter space.

In practice the parameter space must made discrete and parameter k and d are sampled into a limited number of values which represent a rectangular structure of cells. This is called the accumulator array A, whose elements are accumulator cells $A(k, d)$. For each possible line point $P_i = (x_i, y_i)$ in the original image the corresponding accumulator cells are calculated with $d_j = -k_j x_i + y_i$ for each sampled k in the discrete parameter space and are increased by one. If there is a straight line $y = ax + b$ present in the original

image the accumulator cell $A(a, b)$ will be increased as many times as it is detected as a line possibly going through a point P_i. Therefore lines existing in the image can be found by local maxima detection in the accumulator space.

The parametric equation of the line $y = ax + b$ is good for explanation of the Hough transform principles, but it causes difficulties in vertical line detection ($k \to \infty$). A way around is to use the normal representation of a line:

$$s = x \cos\theta + y \sin\theta. \tag{3.5}$$

The algorithm steps are the same as for the slope-intercept representation. Instead of straight lines in the k, d space, each point of the original image leads to sinusoidal curves in the s, θ space [GW93]. Figure 3.10 shows an example of an single detection using equation 3.5. Again the resulting line is found by detecting the local maxima in the accumulator.

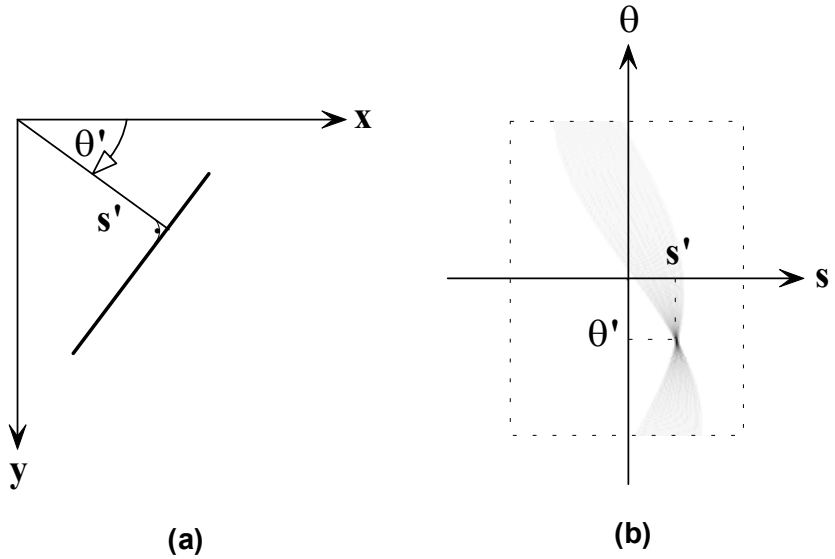


Figure 3.10: Hough transform in s, θ space: (a) straight line in image space; (b) s, θ parameter space.

3.2.2 Detection of Circles

As mentioned before the Hough Transform can be used for detecting of any shapes given by an analytic expression. If circles should be found it is

$$(x - a)^2 + (y - b)^2 = r^2 \tag{3.6}$$

where the circle has center (a, b) and radius r. Therefore the accumulator data structure must be three-dimensional (parameter space a, b, r). Knowledge about the size of the circles

can be used to reduce the accumulator dimension to two (parameter space a, b) which is used in this work. Figure 3.11 shows an example of circle detection with known radius. Each point in the original image (a) leads to a circle in the parameter space (b) and the center of the circle can be found again by maxima search in the parameter space. Figure 3.11 (c) and (d) demonstrate the power of the Hough transform on noisy data - the real center is still found although 30% salt and pepper noise is added to the original image.

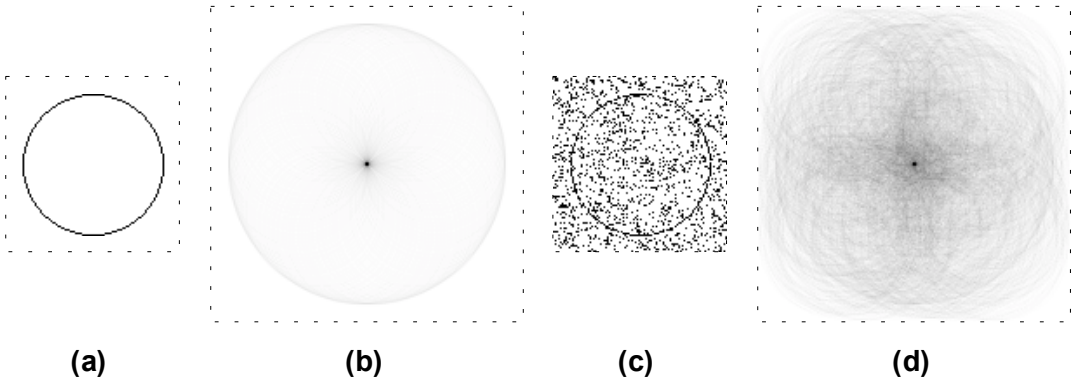


Figure 3.11: Hough transform - circle detection with known radius: (a) original image; (b) parameter space; (c) noisy image; (d) parameter space.

3.3 Thresholding

Grey-level thresholding is the simplest segmentation process to separate image objects from image background. It is the oldest segmentation method but still widely used in simple applications due to it's computational inexpensiveness and real-time capability [SHB93].

3.3.1 Basic Thresholding

Basic thresholding is an transformation of an input image f to an binary output image g depending on an threshold value T such as:

$$\begin{aligned} g(x,y) &= true \ for \ f(x,y) \geq T \\ g(x,y) &= false \ for \ f(x,y) < T. \end{aligned} \tag{3.7}$$

Input pixel with gray-level $\geq T$ are set to $true$ in the output image representing image objects, $false$ output pixel define the image background. If objects do not touch each other and their grey levels are clearly distinct from background grey-levels, thresholding

is a suitable segmentation method. Figure 3.12 shows an example for thresholding with different threshold values. It can be seen that the correct threshold selection is critical for successful threshold segmentation. It is only seldom possible to use a single threshold for the whole image, called global thresholding. Adaptive thresholding is using variable thresholds, which can be a function of local image characteristics or a function of time, for example if light conditions are changing over time and the threshold is adaptive calculated to compensate those light changes.

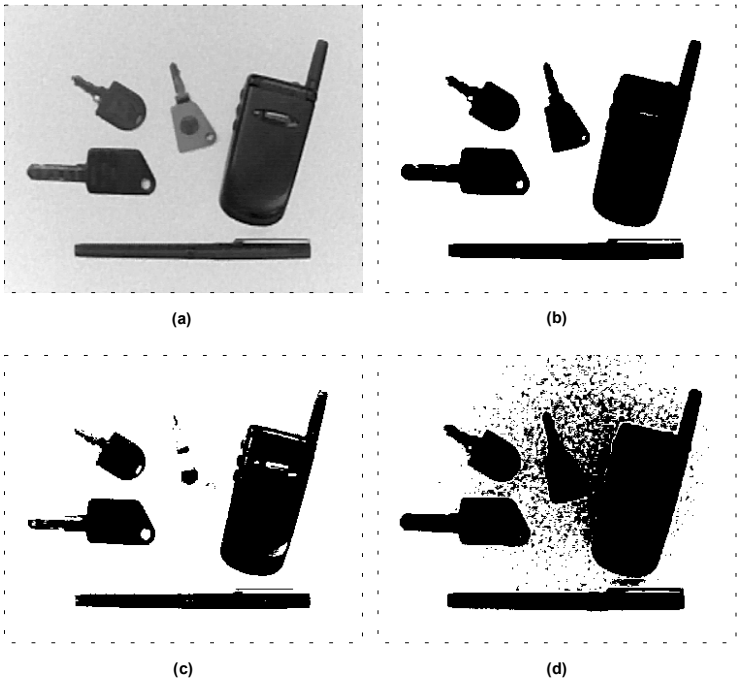


Figure 3.12: Image Thresholding: (a) original image; (b) threshold segmentation; (c) threshold too low; (d) threshold too high

3.3.2 Optimal Thresholding

Thresholding methods based on approximation of the histogram using probability densities with normal distribution is called optimal thresholding. The threshold is set to the closest gray-level corresponding to the minimum probability between the maxima of two or more normal distributions. The difficulty with these methods is the estimation of the normal distribution parameters and the uncertainty if the distribution may be considered normal. This problems may be overcome if an optimal threshold is sought that maximizes the gray-level variance between objects and background [SHB93].

An simple iterative algorithm shows the rationale for this approach [RC78] and assumes that regions of two main gray-levels are present in the image. Assuming no knowledge

about the exact location of objects, a first step approximation takes the four corner pixel in the image as background and the remainder as object pixel. In every step t the mean background gray-level μ_B^t and the mean object gray-level μ_O^t are calculated.

$$\begin{aligned} \mu_B^t &= \frac{\sum_{(x,y)} \in_{background} f(x,y)}{\#background\_pixel} \\ \mu_O^t &= \frac{\sum_{(x,y)} \in_{object} f(x,y)}{\#object\_pixel} \end{aligned} \tag{3.8}$$

$$T^{(t+1)} = \frac{\mu_B^t + \mu_O^t}{2} \tag{3.9}$$

The new threshold value $T^{(t+1)}$ provides an updated background to object distinction for the next step. If $T^{(t+1)} = T^{(t)}$ the optimal threshold is found and the iteration is halted. This method performs well under a large variety of image contrast and brightness conditions which is shown in Figure 3.13, four to ten iterations are usually being sufficient to find the optimal threshold.

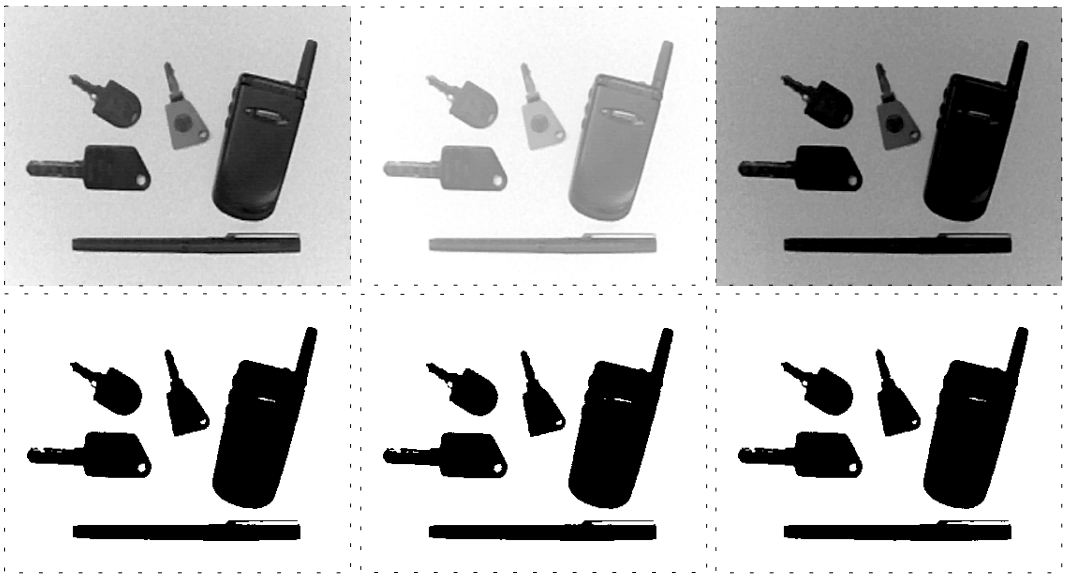


Figure 3.13: Iterative thresholding under variety of image contrast and brightness.

3.4 Convolution

Convolution is an important operation in the linear approach to image analysis. The convolution g of two-dimensional functions f and h is denoted by $f \star h$, and is defined by the integral

$$\begin{aligned} g(x,y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a,b)h(x-a,y-b)dadb \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a,y-b)h(a,b)dadb \\ &= (f \star h)(x,y) = (h \star f)(x,y). \end{aligned} \tag{3.10}$$

Convolution is a very useful linear, translation-invariant operation. A digital image has a limited domain on the image plane, and so translation invariance is only for small translations, it is thus often used locally and expresses a linear filtering process using the filter h, which is often used in local image pre-processing and image restoration [SHB93].

The discrete 2-D image convolution g_d is defined by the relation

$$g_d(x,y) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f_d(m,n)h_d(x-m,y-n). \tag{3.11}$$

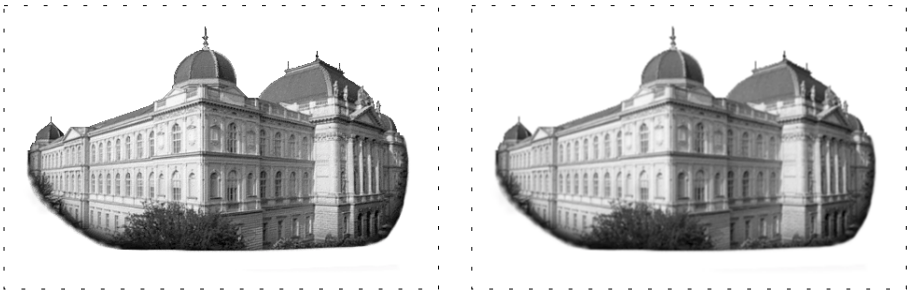
The algorithm can be illustrated by moving the so called convolution kernel h_d over the image and calculate the sum of all multiplied corresponding elements. Depending on the kernel the image is linear filtered, for example smoothed using a Gaussian kernel. Figure 3.14 shows a sample image and the filtered smoothed result using a Gaussian kernel while figure 3.15 shows the result using the horizontal and vertical Sobel kernels for edge extraction.

Practically, computing the discrete convolution in the frequency domain often is more efficient than using equation 3.11 directly. The procedure is to compute the Fourier transforms of the image and kernel by using a fast Fourier transform algorithm. The two transforms are then multiplied and the inverse Fourier transform of the product yields the convolution function [GW93]. Because both the images and used kernels in this thesis are relative small, the direct method is chosen because of better runtimes and no further details about Fourier transformations are given.

3.5 Border Tracing

If a region border is not known but regions are defined in the image, borders can be uniquely detected easily. The image with the region should be binary (*true* regards to the object, $false$ is background). The inner boundary, which is a subset of the region, can be determined using the following tracing algorithm [SHB93]:

1. Search the image from top left until a pixel of a new region is found; this pixel P_0 then has the minimum column value of all pixels of that region having the minimum row value. P_0 is the starting pixel of the region border. A variable *dir* is defined for

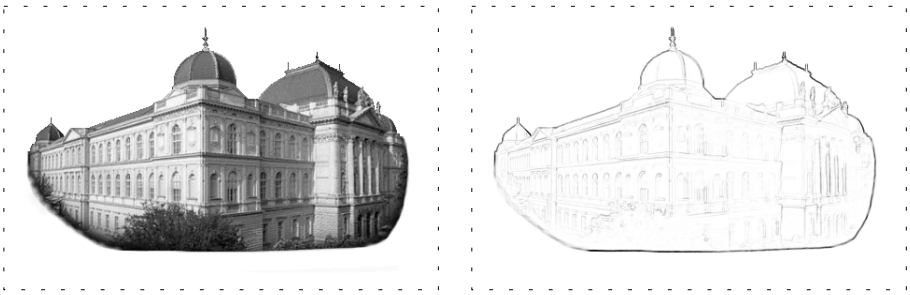


(a) (c)

$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

(b)

Figure 3.14: Image convolution for smoothing: (a) original image; (b) Gaussian convolution kernel; (c) resulting image.



(a) (c)

$$h_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad h_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(b)

Figure 3.15: Image convolution for edge extraction: (a) original image; (b) convolution kernels; (c) resulting image using the addition of the absolut values from the two filtered images using the kernels.

storing the direction of the previous move along the border from the previous border element to the current border element. *dir* is initial assigned to 7 (see figure 3.16 (a)) for 8-connectivity border detection.

2. The 3x3 neighborhood of the current pixel is searched in an anti-clockwise direction, beginning with the pixel positioned in the direction $(dir + 7)$ *mod* 8 if *dir* is even (fig. 3.16 (b)) or $(dir + 6)$ *mod* 8 if *dir* is odd (fig. 3.16 (c)). The first pixel found regarding to the object is a new boundary element P_n. Update the *dir* value.

3. If the current boundary element P_n is equal to the second border element P_1, and if the previous border element P_{n-1} is equal to P_0 the algorithm is stopped, else step 2 is repeated.

4. The detected inner border is represented by pixels P_0 ... P_{n-2}.

This algorithm works for all regions larger than one pixel and is able to find region borders but does not find borders of region holes like contour detection using morphology as shown in section 3.1. Thus, this is intended in some cases, for example for shape comprehension calculations, and because of its low runtime it is often the preferred method for boundary estimation.

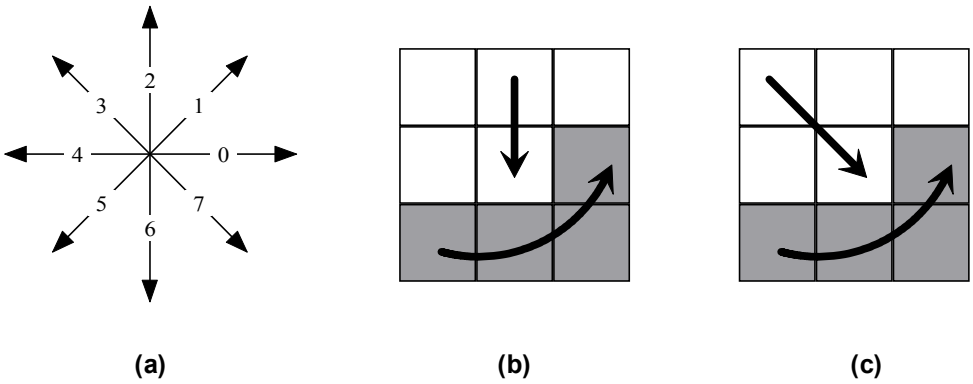


Figure 3.16: Inner boundary tracing: (a) direction notation, 8-connectivity; (b) search sequence for even direction; (c) search sequence for odd direction.

3.6 Region Labeling

Binary images may contain more than one independent regions. Often it is necessary for region description to separate those regions. One method for this is to label each region with a unique number, which is called region labeling. The used algorithm is a standard 8-neighborhood region identification with two passes:

1. First pass: Search the entire image R row by row and assign a non-zero value v to each non-zero pixel $R(i, j)$. The value v is chosen according to the labels of the pixel's neighbors, where only the upper and left neighbors are taken into account:

   - If all neighbors are background pixels, $R(i, j)$ is assigned a new unused label.
   - If there is just one neighboring pixel with a non-zero label, this label is assigned to $R(i, j)$.
   - If there is more than one non-zero pixel among the neighbors, assign the label of any one $R(i, j)$. If those labels of the neighbors differ (label collision), store the label pair as being equivalent to a separate data structure, an equivalence table.

2. Second pass: All of the region pixels were labeled during the first pass, but some regions have pixels with different labels due to label collisions. The whole image is scanned again and pixels are re-labeled using the equivalence table information.

Figure 3.17 shows an example of labeling a binary image showing the labeled result color coded.

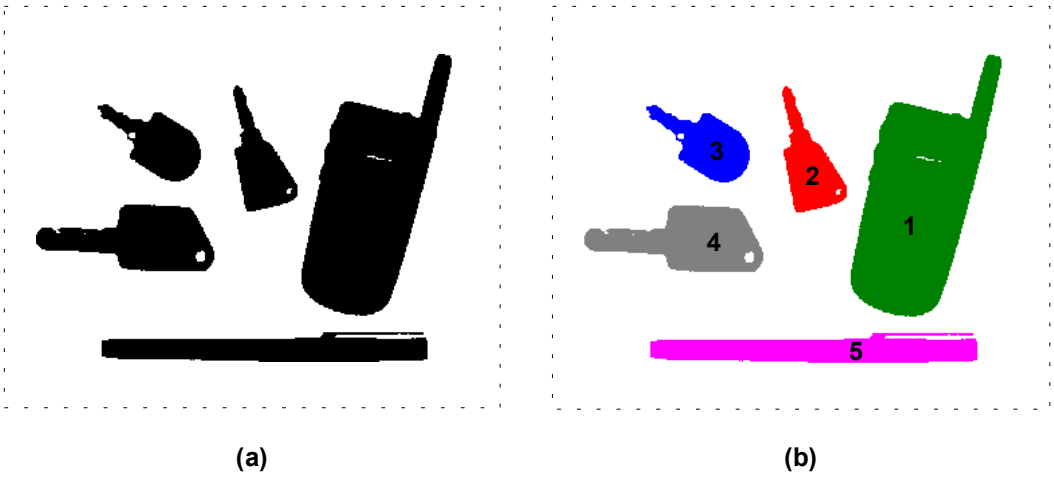


Figure 3.17: Region Labeling: (a) Binary Image; (b) Color coded labeled image.

The upper algorithm can be expanded to get separate binary sub-images instead of the labeled one. Therefore at the second pass for every found label the bounding rectangle around it is estimated and stored. Those bounding rectangles define the sub-images, which are cut out of the labeled image. Those pixel with the right label are set to true while background and pixel with other labels are set to false. The result is a list of binary sub-images containing the separated regions and their offset to the origin of the input image. Figure 3.18 shows this result for the upper example.

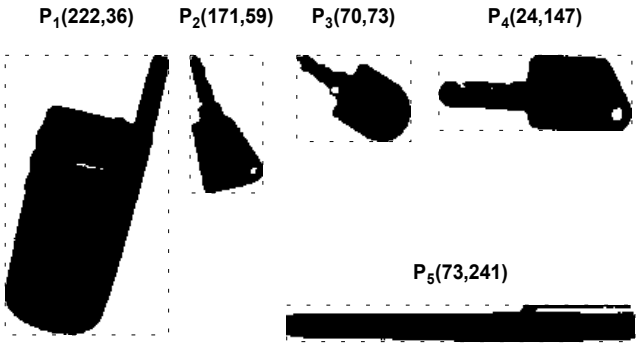


Figure 3.18: Region Labeling into binary subimages with separated regions and origin offsets.

Chapter 4

The Novel Vision-based Approach

In this chapter the new Blackjack analysis system based on real-time image processing is presented and further described. The image processing algorithms which were presented in the last chapter are used in common, defining the vision modules which extract the desired image informations used for the game analysis.

The first section deals with the underlying working environment and the experimental setup which was used. Some details about the image acquisition are also shown. The next section is one of the key parts in this thesis - the description of the used vision modules. Several image processing tasks used to extract the desired information are presented and examples show their benefit. The following section put all components together and show how they work together using logical game-flow controlling. Some details about the prototype interface and user interaction are described as well. At last some further perceptions are pointed out.

4.1 Experimental Setup

The provided resources given by GRIPS Electronic GmbH specify the fundamentals and also restrictions to this work. In respect to commercial usage of the whole system at high quantities in future, the computer and image acquisition components are set on a low cost budget. Furthermore there should be an possibility to append the system components to existing Blackjack tables without great rebuilding and no limitations to the dealer. The basic idea is to add an embedded table computer, a small and unobtrusive camera and an electronic chip-tray to existing tables as shown in Figure 4.1.

4.1.1 Game Resources

An original Blackjack game table including two different game surfaces were provided (see figure 4.2 (a)). The two surfaces were slightly different printed, one with seven and one with nine player regions at different colors. Those different surfaces should be used to work out image processing algorithms which work in a wide spectrum of surfaces, not only at

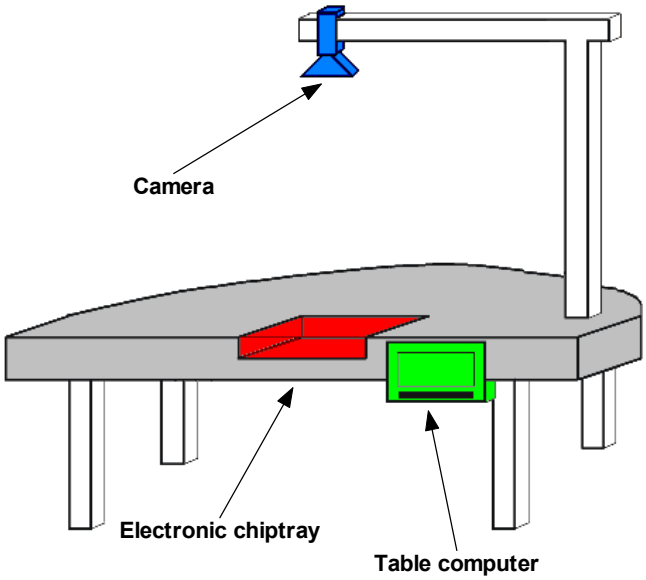


Figure 4.1: Schematic Setup: Table mounted with camera, electronic chip-tray and embedded table computer.

one specific table. Furthermore the regions of a surface should be adaptive definable.

Several diverse card piles and chip stacks are also provided for inspection and analysis (fig. 4.2 (b)). Card and chip sizes, colors and imprints may differ slightly from location to location, therefore an kind of learning mechanism for those objects is needed in order to detect them properly. Details about object learning and classification can be found in section 4.2.

4.1.2 Hardware Platform

The chosen platform for the prototype is a low budget personal computer system (Pentium® II - 400 MHz, 512 KB L2 cache, 128 MB ram, 4GB hard disk) including a ordinary chat cam. There was no embedded system chosen because it should also be used for implementation and project work. The operating system is Windows® 98 because it provided USB support which was needed for first experiments with a USB camera, but the overall implementation is not limited to it, with another user interface and image acquisition model it can be easy adapted to other platforms with little effort.

The electronic chip-tray which replaces the ordinary chip-tray of a normal game table is capable to determine the amount of chips in every slot and can be used to calculate the load and the amount of incoming and outgoing chips over time (Figure 4.3 a). The communication with the controller or a personal computer is done over Ethernet linking (Figure 4.3 b). The implemented prototype is not using the informations provided by the

(a) (b)

Figure 4.2: Game resources: (a) Blackjack table with two game surfaces; (b) various chip stacks and card piles.

chip-tray directly, the events of incoming and outgoing chips are triggered manually to simulate that informations. This was done because the provided chip-tray was old and not working and the overall progress of this thesis should not be slowed down by much interface programming and chip-tray communication handling.

4.1.3 Image Acquisition

First experiments were made with an off-the-shelf Logitech [Log01] USB camera called QuickCam Home (a detailed review of the camera can be found at [Dan98]). The standard TWAIN interface [TWA01] was used, but also after internal coding it was not possible to get any useful results with it. Problems caused in contempt of standards prevented to grab single frame images without using the Logitech user interface, which could not be used because it must be clicked a button for every frame to grab. Also after contacting Logitech no more support could be provided and no dynamic link libraries were made available to use the camera with direct coding.

The chosen alternative interface is called Video for Windows [Vid01] which is an easy to use Windows® standard and supported by many camera manufacturers. The disadvantage with this acquisition system is, that the grabbing of single frames is much more time consuming than with usage of direct driver libraries. Acquisition times vary from 50 up to 300 milliseconds which made this camera not usable for real-time purposes, because in the worst case only three frames per second can be captured and almost no more time is available for computational analysis code.

The general problem is that for this analysis system single frames must be provided in real-time. So it is not possible to record an image sequence and do the analysis afterwards

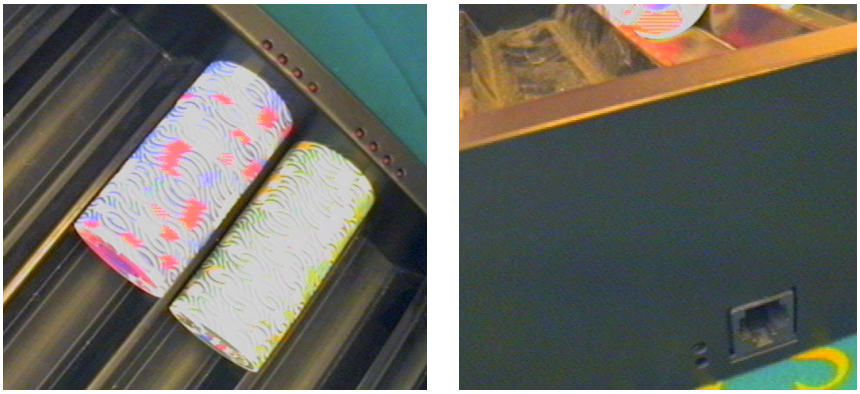


Figure 4.3: Electronic chip-tray: (a) Chip accounting; (b) Communication over ethernet wiring.

as postprocessing. But cheap cameras provide only stream recording to a file which can be done up to 30 frames per second with a resolution up to $640x480$ pixel or slow direct single frame recording where less frames can be grabbed, but the data can be achieved direct. Furthermore it is not possible to adjust shutter, brightness, hue, saturation and other acquisition parameters without using standard interfaces, direct adaptive changes can not be made. This leads to the conclusion that a framegrabber-card with libraries for direct use would be the better choice for real-time applications. Another alternative would be to write libraries for specific cameras, but this is not the field of this thesis.

The compromise between quality and price was found with another cheap chat-cam from Boeder [Boe01], which is using a PCI card instead of USB. It is still without libraries for direct usage, but its drivers for Video for Windows proved better performance and also the image qualities and frame rates are more stable. It supports full PAL resolution ($768x576$) and allows video display adjustment like brightness, contrast, hue and saturation. Those adjustments can only be made using the standard interface dialogs, but answer the purpose for prototyping. The initial thought using such a cheap and low quality camera was that if the analysis system prototype works properly with it, an implementation in real environment must be possible too, and until final implementations the quality for usable cameras will raise while their prices sink.

4.2 Vision Modules

The camera system mounted above the Blackjack table is observing almost the whole table capturing image data for the analysis system. In this section the single-frame image processing modules which are needed to extract the desired analysis information out of the images are described in detail. The first part deals with separation of the input image

in regions of interest followed by segmentation into objects and background. Objects of interest are chip-stacks and card piles, which need to be learned in an initial step (subsection 4.2.3) and detected afterwards during the game analysis (subsections 4.2.4 and 4.2.5).

4.2.1 Regions of Interest

The whole captured image can be separated in different regions of interest. The main regions are the player regions and the card region. For every player on the table there is one region defined (Figure 4.4 - blue areas), where he put down his chips as bet. This are the only interested regions for detecting chip-stacks, other chips may lie in front of the player or beside the player regions for short time if the dealer is exchanging them, but those are not relevant to the game behavior.

The card region (Figure 4.4 - red area) defines the work area for the dealer. There are put down the cards regarding to each player in front of their boxes and his own cards near the chip-tray. Because the dealer cards are playing a prominent role to distinguish game states, a subregion of the card region is defined called the dealer region (Figure 4.4 - dark red area).

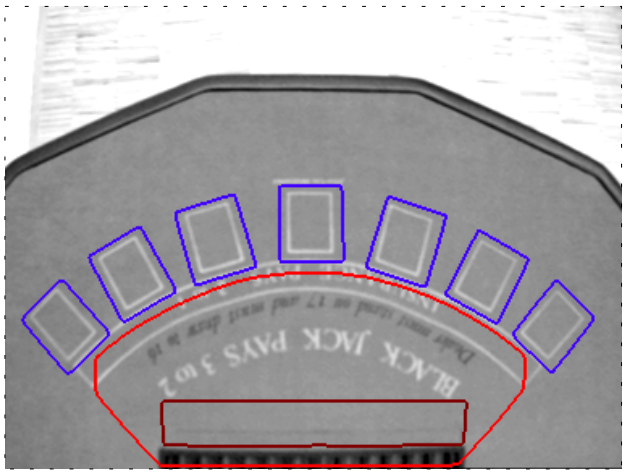


Figure 4.4: Camera view with regions of interest.

Because those regions may vary from table to table an interactive method for setting them adaptive is needed. In consideration of real environment usage it should be possible to define those regions interactive without any computer knowledge in an easy and intuitive way. Therefore paper patterns in the desired size of each region are made, which would be easy to create for different circumstances. To set regions interactive only few steps are needed to perform:

- get an image of the empty table;
- put the player region patterns on the table and get another image;

- put the card region pattern on the table and get another image;
- finally put the dealer region pattern on the table and get another image.

The regions can now computed with low-level image processing algorithms described below. An possible upgrade would be one single pattern with color or texture coded regions where only two steps are needed, but the considerations to find out the regions stay the same for the image processing part.

If the analysis system should also recognize tip bets for the dealer, another region for every player must be defined where chips are identified as a tip. The overall sequence of the game will not change regarding to that and those regions can be treated like player regions where chip-stacks must be found. Thus those tip bet regions are not implemented in this prototype.

Region detection algorithm

A region of interest is a specific part of any shape in the image. To realize it internal a two dimensional area data structure is defined which consists of an binary mask and an offset vector from the image origin to the upper left corner of the mask. The detection algorithm should create such area data out of the captured images for every desired region of interest. The algorithm is described in detail for player regions, it is the same for the card and dealer region. The idea is to separate the region pattern from the rest of the table image, which can be done over subtracting the empty table image from the image with the region pattern. In order to minimize errors caused by camera oscillation the input images are convoluted (sec. 3.4) with a Gaussian kernel to smooth the images. The resulting difference image is thresholded to get a binary image. The used thresholding technique is an iterative approach for optimal thresholding of an bimodal histogram distribution further described in section 3.3.2. The binary image is afterwards morphological closed with a standard cross kernel to smooth the region boundaries and close small gaps or holes in the regions which may occur caused by noise. The last step is to separate all binary objects which is done with a standard region labeling technique described in section 3.17.

4.2.2 Segmentation

The separated regions of interest contain the desired image data which is needed for the game analysis. Methods must be found to determine, if objects of interest are placed inside those regions. Therefore segmentation of image objects and image background is one important step. Segmented objects can be inspected if they are objects of interest, namely cards and chip-stacks. With respect to the desired real-time behavior, the segmentation algorithm must be very fast, but nevertheless robust. It should be nonsensitive to image noise because of the low quality acquisition system. It must also work properly at light and brightness changes, which may occur global at the whole image but also at local image parts regarding to a real world environment in a casino.

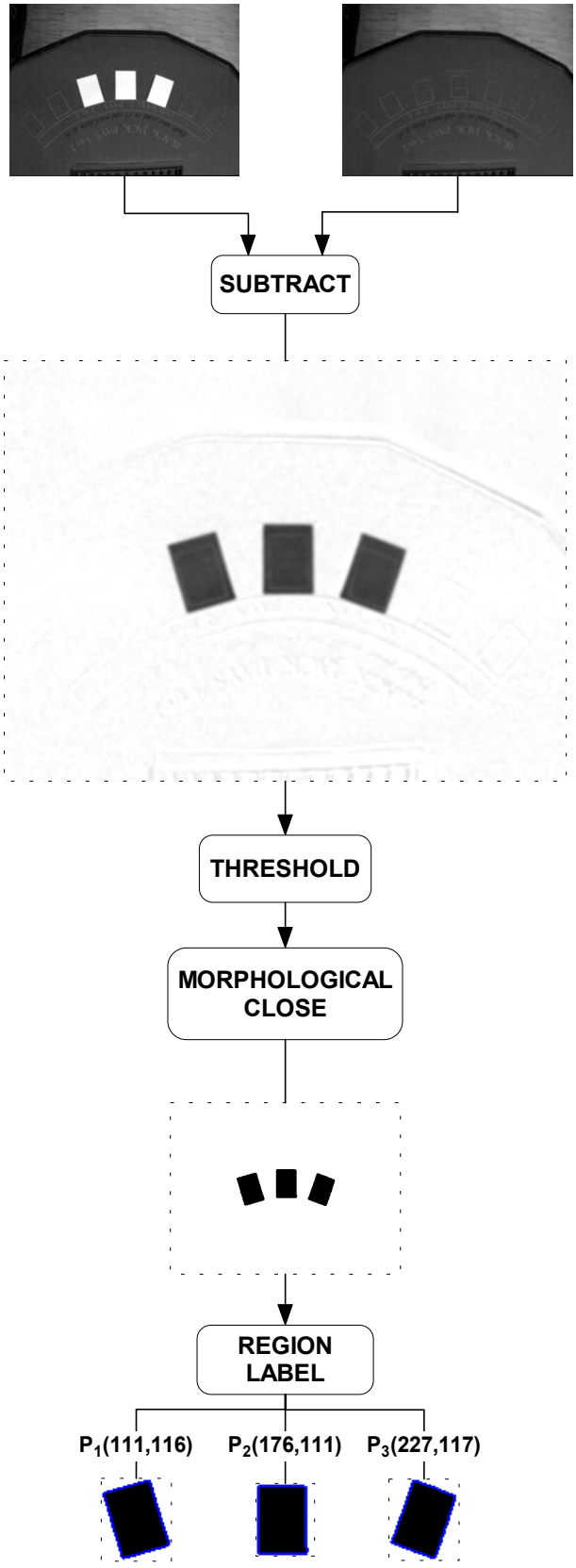


Figure 4.5: Region detection.

Initial Thresholding Techniques

The simplest and fastest segmentation process is gray-level thresholding (3.3). Because chip-stacks and cards are bright blobs on a darker, more or less uniform table surface, thresholding should lead to proper segmentation where the resulting binary image is boolean $true$ for object pixel and boolean $false$ for background pixel. Initial experiments were made with simple global thresholding techniques which lead to problems due to local brightness changes caused by changing light environment. Figure 4.6 (a) shows an example for segmentation of a captured image with global threshold: objects are dark blobs in the resulting binary image, the remaining background is white. Roundish objects can be identified as different types of chips and rectangular objects as cards. Local light changes as seen in figure 4.6 (b) make global threshold techniques useless, parts of the image get over- or under-segmented which means that background pixel are dedicated to objects and object pixel to the background. The solution is to use local threshold algorithms instead of global ones which means that the threshold value varies over the image.

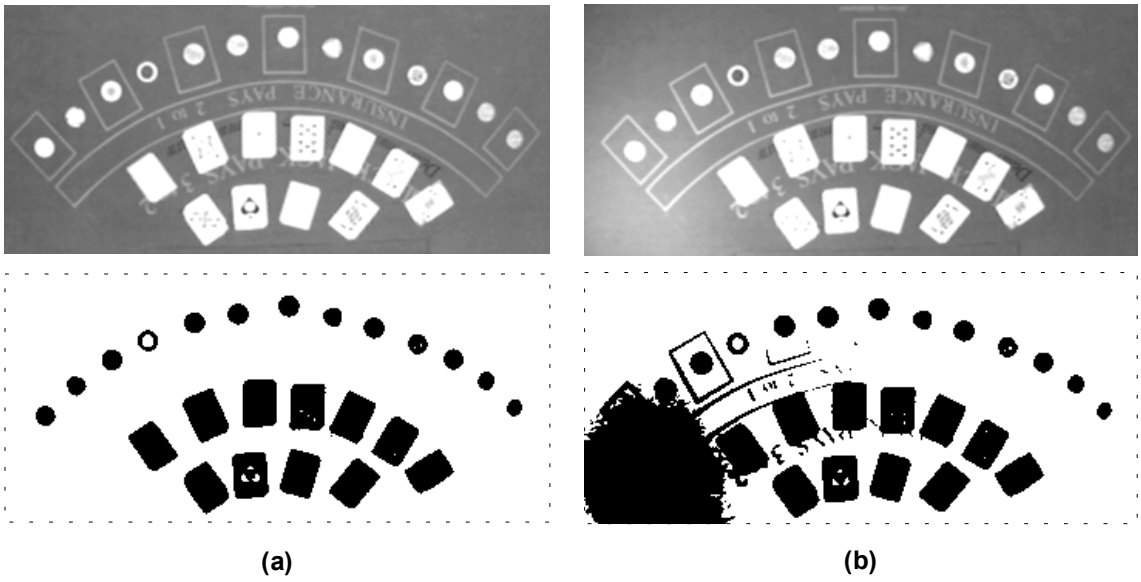


Figure 4.6: Global Thresholding: (a) Uniform light distribution; (b) Local light increase.

An approach to local thresholding can be done with using the former described regions of interest. Every region can be seen as a single image part which is segmented stand alone with different threshold values. Thus, local light differences on the table can be taken into account with using higher threshold values at brighter regions and lower values at darker ones.

The key to proper segmentation with gray-level thresholding is the knowledge about the optimal threshold value, where all object pixel from the original image lead to boolean

true in the resulting binary image and all remaining background pixel boolean $false$. Of course this optimal value it is not existing for all kind of images, especially when object gray-levels and background gray-levels are conflating together. But for the case in this work it should be possible to separate the brighter chips and cards from the darker table surface. The segmentation will not be optimal in all cases, therefore robust object identification methods must be found which can deal with bad segmented data which will be presented in further sections in detail.

The trial and error techniques to determine the desired threshold value used in initial tests is not possible in a real world environment like a casino. Thus, a threshold detection algorithm must be found. The main objectives are still that it is a fast, robust and not sensitive to brightness changes of the input data.

Figure 4.7 shows a single region and its gray-level histogram in various circumstances. Black parts of the input data is outside the region of interest and unaccounted for histogram calculation. The histograms are normalized regarding to their highest peak representing that gray-level, which is most common in the image. The first dataset (a) shows an empty player region where the highest peak is the mostly uniform table surface and the small hill on the right side is the painted region border. Part (b) is the same region with a single chip inside. In the histogram another peak can be seen with the gray-level 255 which represents the chip pixel. This example shows a very simple bimodal histogram and segmentation is easy over setting the threshold value in the zone between the two peaks, for example at value 192, which is shown in part (c). Brightness changes may deform the histogram drastically, which can be seen also in part (c). The former chosen threshold value leads to over-segmentation where the painted surface borders are accounted also as object pixel. A better segmentation is shown also to point out that it is still possible to separate the chip from the background with thresholding.

One method to determine the optimal threshold is an iterative algorithm [RC78] described in detail in section 3.3.2. It is based on the rationale for approximation of the image histogram using two probability densities with normal distribution and assumes that regions of two main gray-levels are present in the image. In this case the histogram is mainly bimodal and the algorithm should be working well for separation of the two main gray-levels, namely chips and table surface. Practical experiments clarified that the algorithm works only properly with some restrictions. It handles brightness changes in a wider spectrum than simple thresholding, only in extreme samples of brightness the painted region borders on the table are declared as object pixels too, which is not intended. Figure 4.8 shows such brightness dependent changes of the segmentation quality of a single chip in a painted player region on the game table. When surrounding light sources increasing the image brightness to a critical level, segmentation errors occur.

At that high intensities the gray–levels of objects and background are fusing together and no thresholding algorithms can be used anymore for proper segmentation. This is mainly caused by oversteering the acquisition system. A possible solution for this problem is found by keeping the camera in its field of work which can be done by using a small initial shutter time which lead to darker images at normal light environment.

Beside the mentioned dependency on brightness changes the general disadvantage of

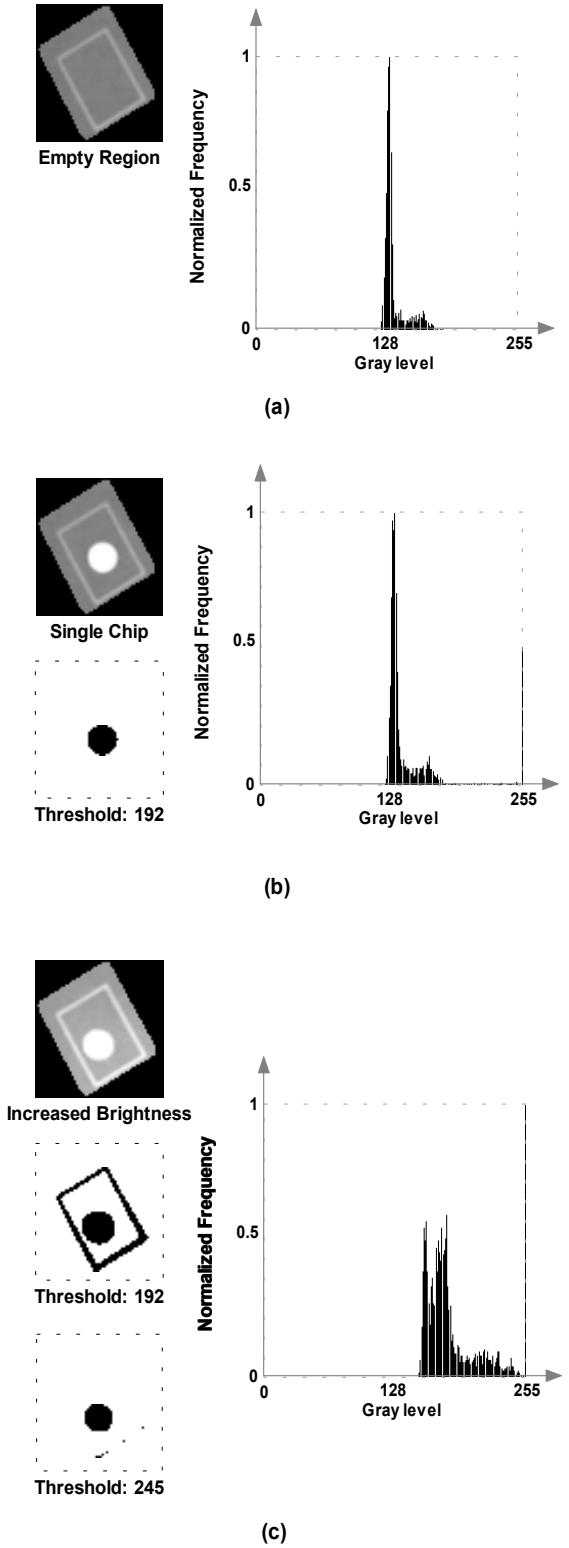


Figure 4.7: Region histograms: (a) Empty Region; (b) Region containing one chip and its thresholded image; (c) Region containing one chip with increased brightness, thresholded images with original and optimal threshold values.

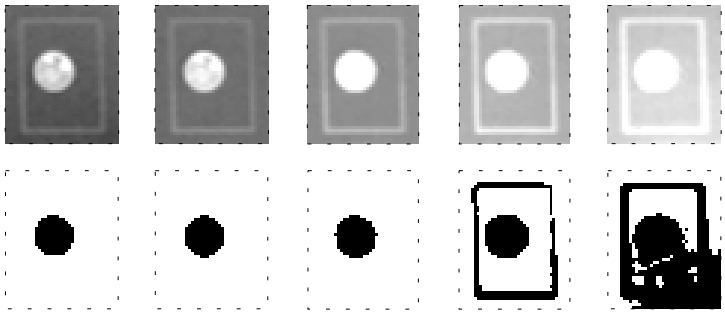


Figure 4.8: Iterative thresholding of a region containing a single chip at various light environment.

the iterative threshold detection method is the following. The initial assumption for this approach was an image containing background and object pixel. Lets suppose that there is no desired object in the image. Guided by a wrong supposed condition the algorithm will try to separate the background into object and background pixel. Accordingly there may be a difference between desired objects and object pixel in the term of optimal segmentation which leads to the conclusion that this iterative approach would not be a good choice for segmentation because there are not always desired objects in the specific player regions. 4.9 (a) shows the proper functionality of thresholding regions with increasing amount of chips while (b) points out the problem with empty regions: in the first case the painted region border is treated as object, in the second one the mainly uniform background is separated in object and background pixel.

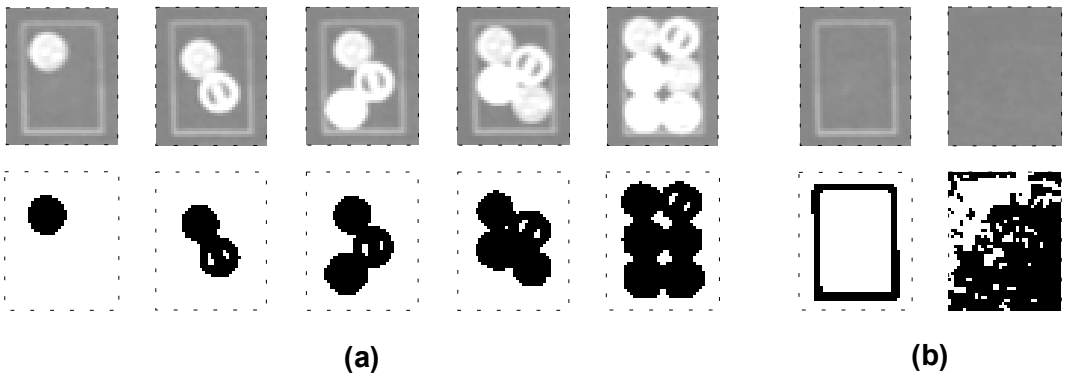


Figure 4.9: Iterative thresholding: (a) increasing number of chips; (b) mis-segmentation of empty regions due to wrong assumtion.

Adaptive Histogam-based Thresholding

The final used segmentation algorithm for this prototype is an adaptive histogram-based approach. For each region an reference histogram based on a captured image from the empty table surface is computed. This step can be made on demand as an initialization step, thus it can handle different table surfaces easily. In this initialization step the initial threshold values for every specific region are computed out of their histograms. Furthermore in every capture step new region histograms are computed. The basic thought of this method is to compare the reference histograms with the current ones. With peak and valley analysis a presumption about the brightness shift is made and the threshold is adjusted regarding to that.

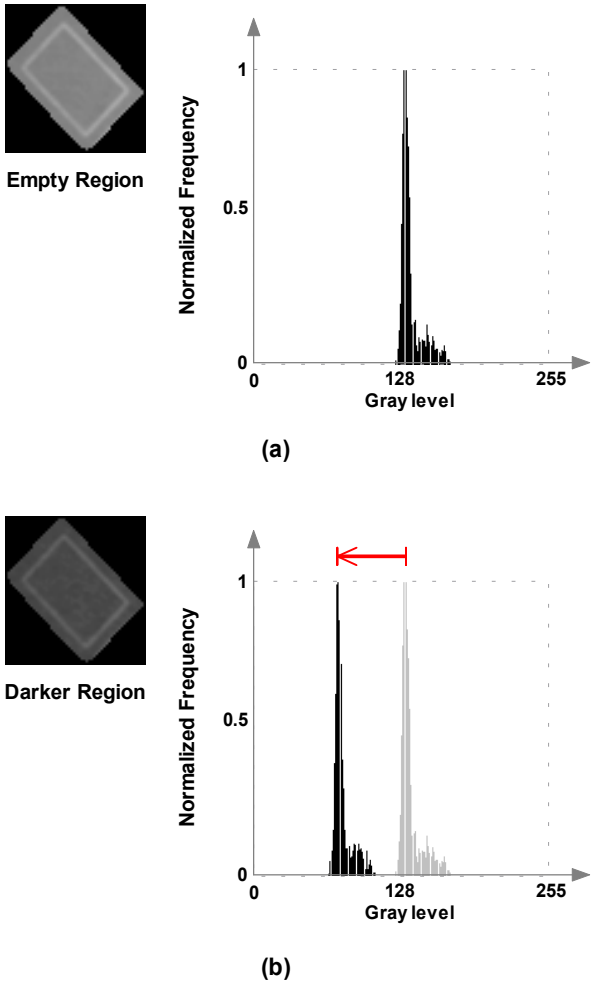


Figure 4.10: Brightness shift and histogram impact: (a) empty region; (b) darker region with shifted peak to darker gray-levels.

Figure 4.10 demonstrates the histogram impact on brightness changes of an empty

region. The highest peak representing the most uniform table surface shifts to darker gray-levels if surrounding light gets trimmed down. The adaptive step is to determine the highest peak in the reference histogram and in the presently one. The difference is added to the initial threshold value to compensate brightness changes. The initial threshold value is derived from the position of the lowest valley right of the maxima representing the table surface. This value is usually the brightest background pixel of the empty region, so thresholding with this value will lead to an empty binary image which is a correct segmentation. This search of the threshold value is needed because it cannot be determined directly out of the brightest background pixel. If just one pixel is brighter than the desired objects which should be separated, the threshold value would be too high and no proper segmentation is possible (brighter pixel are mainly caused by noise). Also direct calculation of the initial threshold value from a reference chip is not possible, because different regions may be in different light environment even at the initialization step. Thereby a global initial value will occur in different segmentation results regarding to the region. On the other hand an empty region may be segmented into some small background artefacts if the threshold value is set to the lowest minima. But in practice this artifacts are occurring infrequently and can be reduced by morphological post processing after the segmentation (see section 4.2.4).

If the two overlaid histograms in 4.10 (b) are compared exactly, it can be seen that changing light environment also changes the shape of the peak slightly. This originates in the non ideal lightsources and reflection parameters of a real world environment. The shape of the peak will change even more if there are objects put down into the region and the overall amount of background gray-levels will be reduced the more objects are put down. This could carried on until more object gray-levels than background gray-levels are in the region and the highest peak in the histogram is not anymore corresponding to background. This extreme case is occurring only rare in practice, nevertheless it must be taken into account. This can be done by using the position of the histogram maxima: if it is near the highest gray levels it must correspond to chip stacks or cards because they are the brightest objects in the image. In that case the next minima left from this peak is computed and used as current threshold value.

To make extrema search in the histogram more robust and also to assess the wider gray-level peaks of the background more than small peaks of chips the histogram is smoothed by convolution (sec. 3.4) with an oblong Gaussian kernel shown in 4.11. It can be seen that the small maxima representing the chip pixel in the original image is suppressed and the wide gray-level spectrum of the background leads to a new maxima in the smoothed histogram.

This presented adaptive method works well in the prototype environment and fulfill the requirements to separate cards and chips from the table background very well. It handles brightness changes in a wide spectrum and can be used with almost full filled regions with objects and also with empty regions. Due to its computational cheapness it can be used in this real-time demanding application.

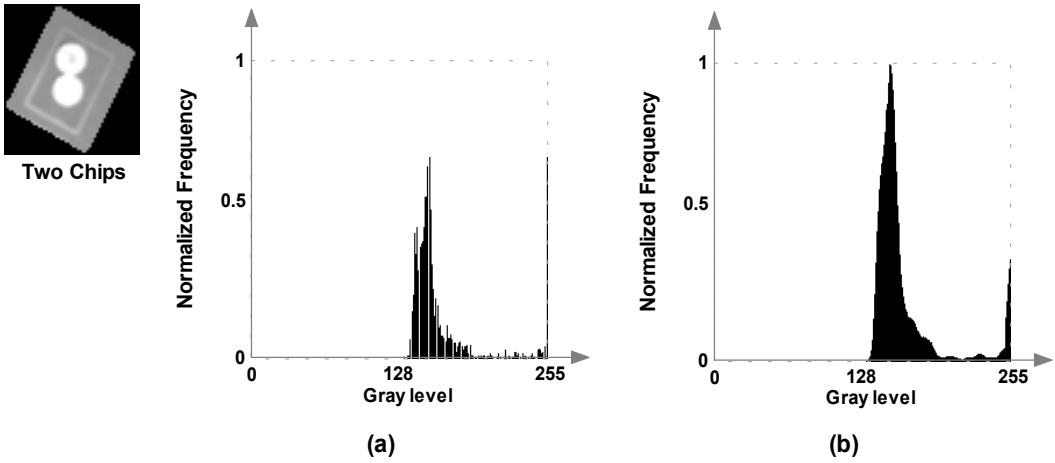


Figure 4.11: Histogram smoothing: (a) computed histogram; (b) convoluted with oblong Gaussian kernel.

4.2.3 Learning the Objects of Interest

Object recognition is still an unknown field of science. A lot of work is still carried out in this field and it is also subject of ongoing work on the institute [BPPP98]. It is not possible to compare each object with a reference object on pixel base directly, object features must be calculated for comparison. Over that features the segmented objects can be classified. The objects required for the analysis are chip stacks and cards, which must be detected in a robust method. Other obstacles like the card shoe or player hands on the table should not affect the system and must be filtered by the object classifier. The chosen features must be working robust with noisy low quality data, the main task is the distinction of chips, cards and hands, which are the most occurring objects on the game table.

Basically shape features can be divided into contour-based and region-based features. While contour-based features are better suited for complex shapes or for borders expressed in some mathematical form, region-based features are limited to deal with simple shape primitives [SHB93]. Concerning the detection of chips and cards, which may be seen as simple filled circles and rectangles, region-based features are the better choice for object description in this field of work. Another advantage of region-based features is the better behavior with noisy data, where artefacts often lead to wrong contours while regions are manipulated only slightly.

Chip Features

Blackjack game chips are round platy objects which may vary in size and imprint from location to location. By their surface composition properties in comparison to the game table surface they are reflecting light much more and are brighter objects in a captured image. At least a bright visible border must be present to differ them from the background.

Compactness is a popular shape description characteristic which is given by

$$compactness = \frac{(region\_border\_length)^2}{area}. \tag{4.1}$$

Because of the circular shape of the chips compactness is used as a first approach for classification of a single chip. The most compact region is a perfect circle with its compactness value

$$c_{circle} = \frac{(2r\pi)^2}{r^2\pi} = \frac{4r^2\pi^2}{r^2\pi} = 4\pi = 12,566.$$

Circular chips should be very compact too, so their compactness value must be in a short interval around 4π, while cards and hands should have higher values:

$$c_{quadrat} = \frac{(4a)^2}{a^2} = \frac{16a^2}{a^2} = 16.$$

Some examples of arbitrary shapes and their compactness are shown in figure 4.12. In part (a) various shapes of binarized single chips are used for compactness calculation. Their almost circular form lead to values around 4π. The occurrence of values, which are smaller than the compactness of an ideal circle is caused by the rectangular pixel based raster and the improper calculation of object areas and border lengths regarding to it. Part (b) shows rectangular objects, a card and a hand which have a much higher compactness values, which means that their compactness is smaller.

The implemented calculation method is using a modified border tracer (see section 3.5 for detailed description of border tracing) to get the region border length and the sum of area pixel. The tracer adds up the inner border pixel of the object and weight diagonal border pixel with $\sqrt{2}$, because the effective length of one pixel over a diagonal border is its diagonal in a rectangular raster grid. If that would not be taken into account problems with the compactness of rotated objects will occur: the amount of border pixel of a rectangular object as shown in 4.12 (b1) and the amount of border pixel of the same, but rotated object for 45 degree (4.12 (b2)) will change but the amount of area pixel will stay the same. This would lead to different compactness values for the same object depending on its angular position.

Because compactness is independent of linear transformations the diameter of the circle is added to the classifier. As it can seen in 4.12 (c) the compactness can only be used for single chip classification, the values of compounded chip stacks are varying too much and cannot be discriminated from other objects. To detect also compounded chip stacks, a so called chip mask is added to the classifier. This chip mask is a binary image of a reference chip which is computed in an initialization step to stay adaptive to different chip sizes. An example for a good chip mask is 4.12 (a4). This mask is used also for chip detection with an iterative clipping technique which is described in section 4.2.4.

Summarizing the above, chip stacks are identified by three extracted features: diameter, compactness and binary chipmask. Reference features are calculated from an single chip

in an initialization step and are used to classify objects as chips in the detection step of the analysis.

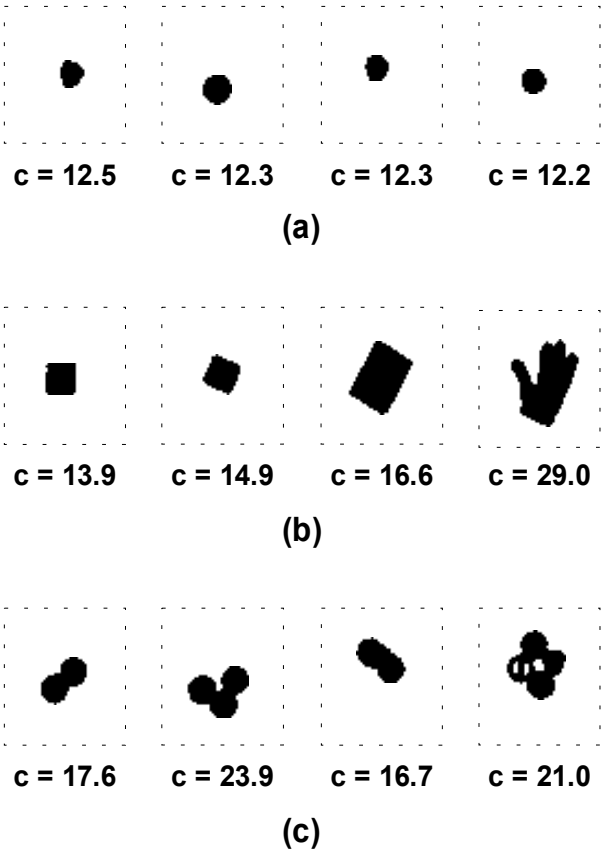


Figure 4.12: Compactness of abitrary shapes: (a) various single chips; (b) recangular objects like cards; (c) compound chip stacks.

Card Features

Blackjack game cards are rectangular with standard size, but the overprint is mainly unique for every casino. The colors, symbols and digits on them are very different from location to location. Nevertheless they are brighter on captured images regarding to the table surface because of their reflection properties like chips. Interior parts of the cards may be darker than the table surface, but at least bright borders can be seen for segmentation.

As a first approach for card identification some of the chosen features for chip detection are also used for cards. The compactness can be computed in the same way, area and border pixel as well. But those features are not sufficient, hands and other obstacles may have almost the same compactness as cards, which was mentioned in the last subsection. Further properties of the cards are used, namely its length, width and diagonal. Those

features can be extracted out of the object border, which consists of four straight lines in the usual case. Out of that border lines the corners can be computed and finally the desired dimension features.

One difficulty to deal with is that cards may be overlapped partially in several different kinds. Figure 4.13 shows some typical dispositions, often straight overlapping ranks are build, but also slant and fan dispositions may be used. The fact, that cards are always put down on the table one by one can be used to simplify the detection of more overlapped cards. It will be shown that only the detection of one single card and two overlapped cards is needed for proper game analysis. The used algorithm to extract the additional card features is based on the Hough transformation, detailed informations can be found in section 4.2.5.

Figure 4.13: Typical card dispositions on the table surface.

4.2.4 Chip Detection

One of the vision key tasks is the detection of single and compound chipstacks in each captured image. This detection is done by comparison of chip features (explained in section 4.2.3), which are extracted from every segmented object, with the specific reference features. In this section the main algorithmic for initialization and further detection is described in detail.

Initialization

As mentioned before, specific reference features of chipstacks need to be extracted for later comparison. To be adaptable to different locations where chip sizes or their imprint changes, the reference features are not pre-determined and hard coded, they are extracted on demand. For this purpose a single chip is put down on the empty table surface and a

reference frame is captured. Out of that image containing only this specific chip and the image of the empty table surface the chip features are extracted. The main parts of the used algorithm are:

- difference image calculation
- thresholding of the difference image
- region labeling
- threshold adaptation
- diameter extraction and synthetic chip creation
- compactness calculation.

The two input images, one of the empty surface and one containing the single chip, are convoluted (sec. 3.4) with an Gaussian kernel in order to smooth the chip boundaries to get a better binary approximation of the shape but also to reduce the influence of small vibrations of the acquisition camera, which may lead to a small displacement of the two captured images. The image of the empty surface is afterwards subtracted from the image containing the single chip. The received difference image contains mainly the difference of the brighter chip gray-level to the table surface gray-levels, but also small gray-level differences caused by noise or small displacement differences of the table surface.

The histogram of the difference image is computed which is the base to determine the threshold value to segment the difference image. The highest peak of the histogram is corresponding to the overall difference of the surface caused by brightness changes and noise while the highest gray-levels correspond to the chip. Usually the highest peak is near zero (see figure 4.16 (a)), but it may shift to higher gray-levels if brightness changes are too high (b). The threshold value is set in the centre of the interval between the highest peak and the highest gray-value to get a first separation of the chip from the background. Reasonable results are achieved even if the light environment changes within the initialization step. To keep those light environment changes low, the time between the capturing of the empty table and the single chip should be kept short.

The resulting binary image is separated in distinct blobs with region labeling (sec. 3.6) where small artefacts get filtered using a size threshold and only one blob remains which is the segmented chip. In order to enhance the quality of the extracted binary copy of the chip another adaptation to the threshold value is done. The binary region is dilated (sec. 3.1.1) with an standard cross kernel to enhance and smooth its borders. The final threshold value is calculated directly from the convoluted image containing the chip. It is set to the mean value of the gray-levels of those pixel, which are masked by the first approximation of the binary chip. The image is thresholded again and the new binary chip is cropped out from this binary image, the position of the chip is known from the first approximation. At last, possible holes within the binary chip which may be caused by non-uniform chip brightness are filled. This binary chipmask is used now for first feature extractions.

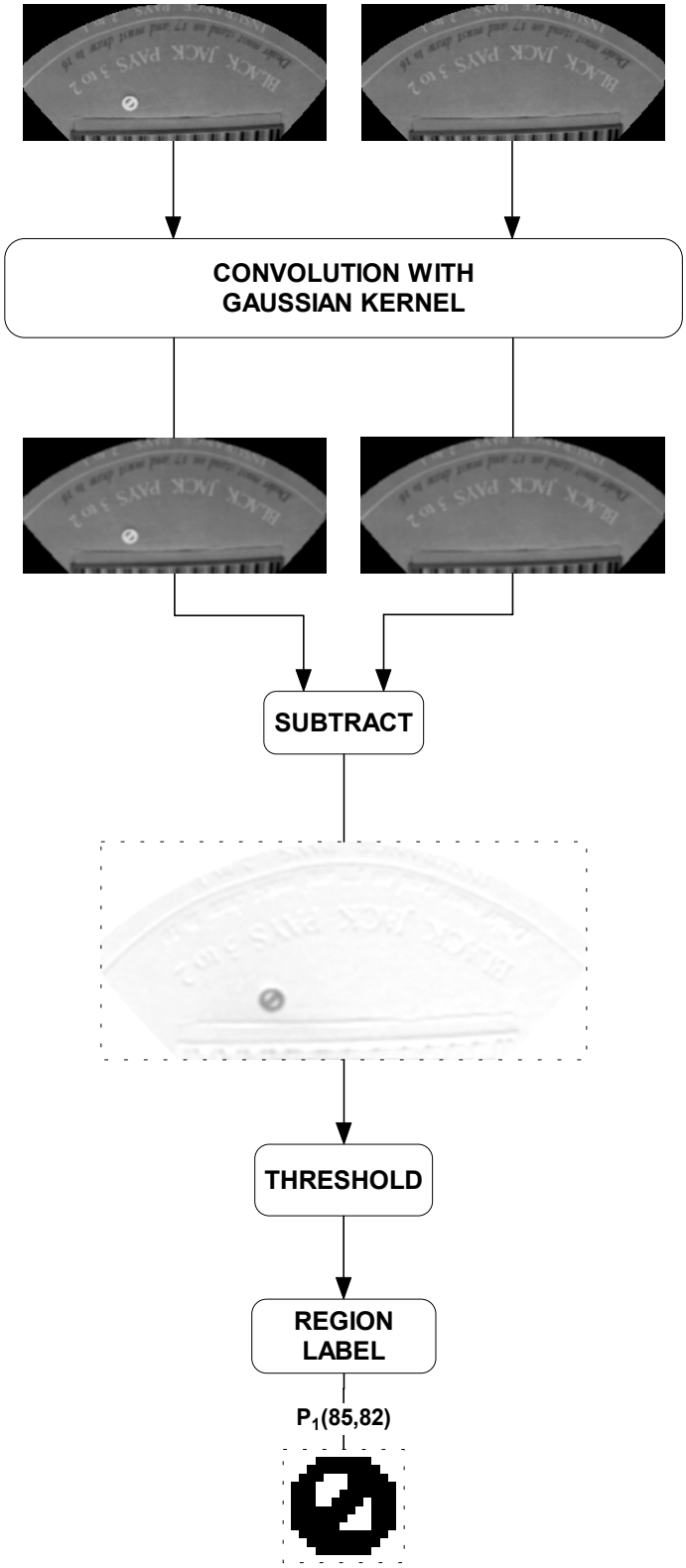


Figure 4.14: Initialization for chip detection part one - first threshold approximation.

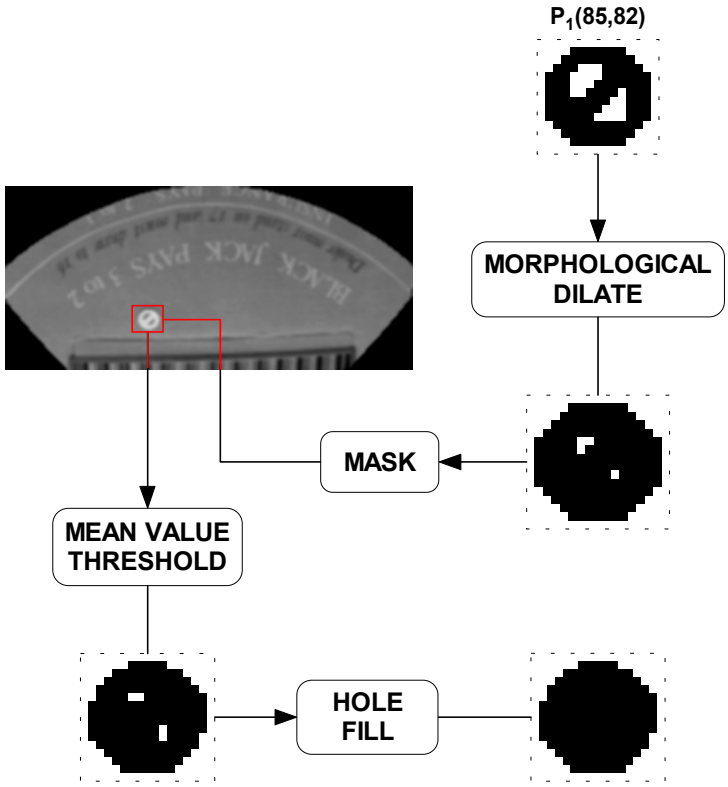


Figure 4.15: Initialization for chip detection part two - threshold adaption and hole filling.

Its compactness (see section 4.2.3) is calculated and checked if it is within the tolerance interval of circular objects. If this check fails the blob is rejected as reference chip and the initialization must be done again. In practice this occurs only caused by user errors, for example trying to use a card or his hand as reference chip.

The diameter is calculated from the width and height of the allowed blob, which is used for synthetic reproduction of an ideal binary chip. This is a filled circle with the calculated diameter which is the used chipmask for the further detection steps. The additional chip features used for detection are also extracted from this mask, namely the pixelsum and the compactness of this chipmask are recalculated and used as reference data.

This more complex thresholding technique with the difference image and second adaptation of the threshold is used in order to enhance the quality of the reference data and robustness. The used position for the chip initialization is the card region, which is in the center of the table, to have lowest possible distortions caused by the camera. The difference calculation is used because of the miss-segmentation of bright table surface overprints at usual thresholding techniques.

Another task of the initialization step is to create the reference histograms for all empty player regions. As described in section 4.2.2, the histograms are calculated from the respective regions, they are smoothed by convolution with an oblong Gaussian kernel

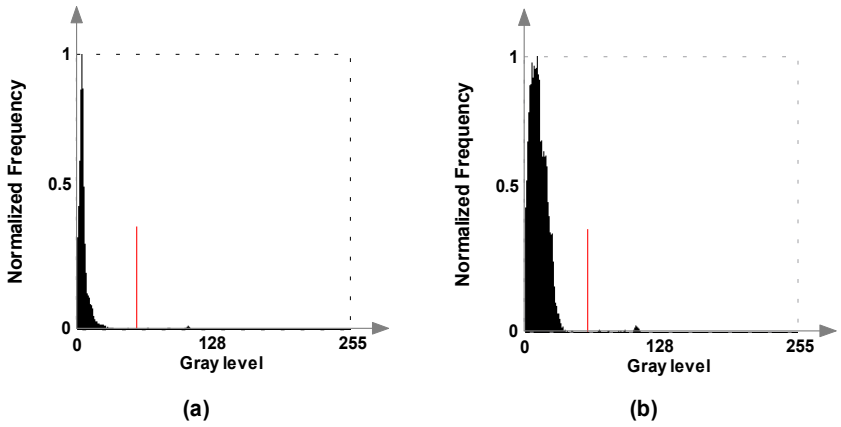


Figure 4.16: Difference image histograms: (a) small differences mainly caused by noise; (b) additional differences caused by light environment changes.

and the highest peak and the first lowest minima to the right of the peak are determined. After this the system is ready to detect single and compound chipstacks in the following detection steps.

Detection

The detection of single and compound chipstacks is done for every captured frame after the initialization is finished. It is a single frame image processing chain which handles every frame independent. The desired output of the detector is the amount of chipstacks in each player region of a specific frame. The pooling and evaluation of the achieved data of each frame is done by a controller which handles the logical game flow (section 4.3.2) and the image stream dependencies (section 4.3.1).

The detection algorithm can be split up in four main parts: segmentation, artefact reduction, feature extraction and finally classification. Each frame is divided into the desired regions of interest, and each region, where chips should be detected, is further processed. The prototype is using the player regions only, but it is a simple task to add other regions where chipstacks should be detected, for example in the card region to detect chip exchange or separate tip bet regions to detect dealer tip bets. Those regions are binarized with the histogram-based threshold algorithm presented in section 4.2.2. Because of miss-segmentation caused by region borders or noisy data the binary images are morphological processed before following feature extraction steps. Small artefacts are removed by morphological erosion (see section 3.1.1) with a standard cross kernel. The deformation of the remaining blobs is revoked by morphological dilation. To prevent blobs from growing too big, a modified version called geodesic dilation [Soi99] is used. The difference to normal dilation is, that an additional binary mask is used. The dilation process is restricted to this mask, even if it is applied more often. This is used to reconstruct small boundaries which may break after the erosion but need more then one iteration of dilation

to be reconstructed.

The binary images are then separated in distinct blobs, which are further used for feature extraction and classification. Usually only one single or compound chipstack will be found in a player region, but sometimes more players are using the same region and so more independent chipstacks may be found within one region. The separation process is done with enhanced region labeling: only those blobs which are not connected with the region border are labeled, others are abolished. This can be done because such blobs are not chipstacks, but hands reaching into the region. Those don't need to be analyzed any further and can be neglected, which reduces the computational costs as well. To ensure that chipstacks in the player regions don't nudge the region border, the regions of interest are set to a bigger size than the original painted player regions on the table.

Every separated blob is now further processed and the features for classification are extracted. To calculate its proper compactness the binary blobs holes need to be filled. This is done by region filling of the outer blob border and boolean X-NOR'ing the resulting binary image with the original one. With this fast technique all inner holes of the blob are filled without determination of their location. After this filling the right pixelsum and compactness are calculated. Basic chip classification can be done with the achieved data:

- pixelsum < reference pixelsum - tolerance -> no chipstack
- pixelsum > pixelsum threshold -> no chipstack
- compactness > compactness threshold -> no chipstack
- compactness in reference compactness interval -> one single chip.

It is obvious that tolerances and intervals are needed for classification, a simple value comparison is not possible in a real world environment. The tolerance and interval ranges are determined during experiments and are mainly dependent on the image acquisition system, the image resolution and the level of noise. The usage of automated or learning classification methods was renounced because of the simpleness of the needed classifier and the easy to understand dependencies.

The pixelsum threshold is defined as the maximal amount of allowed compound chips multiplied with the pixelsum of the reference chip. The usage of this threshold is mainly to save computational costs for the prototype, to fulfill the problem specification four compound chipstacks must be detectable.

The compactness threshold is defined as the maximal possible compactness value reached by any formation of compound chipstacks. The value is determined by oblong dispositions of chipstacks which have the highest possible compactness value, which is again dependent of the amount of the maximal allowed compound chipstacks.

The main part of occurring situations in practice can be classified with this simple extracted features and interval ranges very fast. Not uniquely classified blobs are further analyzed with the usage of the binary reference chipmask. The basic idea is to use the binary chipmask for iterative blanking of the blob by boolean X-OR computations as shown briefly in figure 4.17 (f).

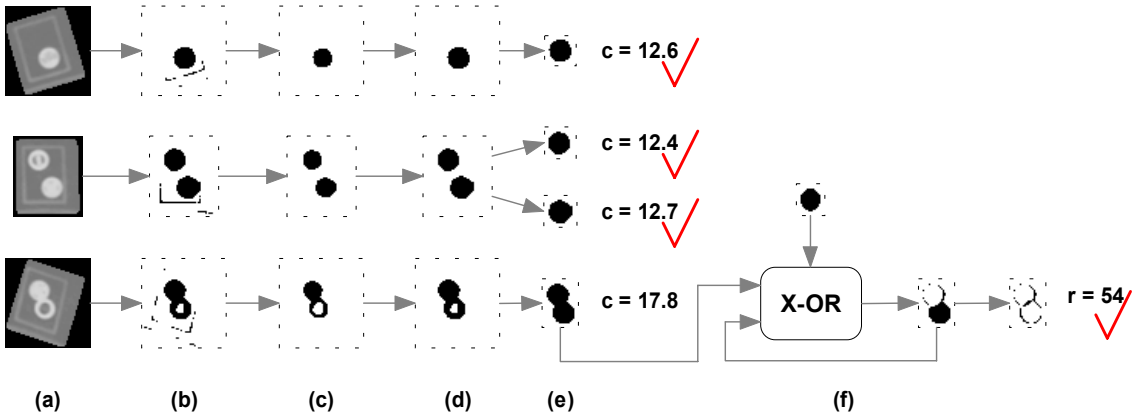


Figure 4.17: Detailed view of chip detection steps: (a) player region; (b) binarized image; (c) eroded image; (d) reconstructed image with geodetic dilation; (e) region labeled and hole filled independent blobs; (f) further classification with iterative XOR-computations with the chip-mask.

Boolean X-OR Classification

Not all blobs can be classified uniquely with the simple extracted features. Dispositions of chipstacks may have almost the same pixelcount and compactness as other obstacles, fingers, even cards. Thus, an additional feature is needed. Because the shapes of compound chipstack dispositions are very different, no classifier can be found which handles a blob as a whole. But it is assembled out of single chips, which shapes are well known. Thus, the approach to detect compound chipstacks is to use the shape of one chip and check, how well and how often this shape fits into the whole blob. Therefore a chipmask is needed, which is created synthetically in the initialization step and represents the reference shape.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.1: Boolean X-OR

The concept of the algorithm is that the chipmask is moved over the blob and searches the position of the best fit into the blob. At this position the chipmask shape is cropped out from the blob. This is done iteratively and a criterion for classification is the amount of remaining pixel from the blob.

The boolean X-OR operation showed the perfect behavior for this cropping work: let suppose the blob represents a finger, pen or any other object, which is thinner but longer than a single chip. If the chipmask is just used for iterative subtraction from the blob,

there will be no pixel remaining and the system classifies the blob as a compound chipstack, which is wrong of course. But if boolean X-OR is used, those parts of the chipmask which fit into the blob will be blanking those pixel out of the blob while the other parts of the chipmask which are not fit within the blob shape will be added to it. Thus, the blob will result in a larger amount of remaining pixel and can be proper classified. Larger objects like cards or hands are processed the same way, and because the chipmask will not fit perfectly into the blob many remaining pixel will occur. On the other hand if a blob of compound chipstacks is processed, the iterative steps of the algorithm will crop one chip after one out of the blob, only few pixel will remain and the object is classified as a compound chipstack. The number of combined chipstacks is determined by the number of iterations which are needed to get a minima of remaining pixel. In figure 4.18 various shapes (two chipstacks, three chipstacks, one card, a hand and a bunch of keys) are processed by this algorithm and the amount of remaining pixel is calculated for classification. An advantage of this iterative algorithm is the adaptive conformation of the amount of remaining pixel. The threshold for one chip can be set to a very low amount and is increased with the number of iterations, which improves the accuracy of the classification. The behavior of the boolean X-OR operation can be seen in the last iteration steps of the hand and the keys samples very well: those pixel of the not fitting parts of the chipmask are added to the object.

The success of this approach depends mainly on the estimation of the right position, where the chipmask fits best into the object. The first criterion which can be used for determination of the best position is the amount of remaining pixel of the resulting binary image after the X-OR computation with the chipmask. The basic thought would be to shift the chipmask over the whole binary image and calculate for every position of the chipmask on the image the remaining pixel from the boolean X-OR operation. But that alone is not working proper because of ambiguities. The problem is, that more than one positions may have the minimum of remaining pixel and one specific must be chosen, but it can also happen that the best position is not the one with the minimum of remaining pixel, which may caused by small image distortions or noise. Some examples are shown in figure 4.19, the amount of remaining pixel is calculated for every position of the moving chipmask (the origin of the chipmask is set to its center), darker pixel indicate lower remains, the regions of minima are marked red. It can be clearly seen that more position can be found which will result to a minimum of remaining pixel after X-OR'ing the chipmask. The amount of positions increases with the size of the object shapes, which is not astonishing because at bigger objects there is more space inside for a full fit of the chipmask.

The determination of the proper position within those regions of minimal remaining pixel is very important, wrong estimation may lead to big errors and even missclassification. Figure 4.20 (a) shows an example of bad positioning: for this iteration the position will lead to minimal remaining pixel, but in further iterations overlapping pixel will be reproduced again by followed X-OR operations while figure 4.20 (b) points out optimal positions, only few pixel remain after three iterations.

The estimation of the position is taking the distance from the region corners into account. Positions which are near to any region corner are preferred and a successive processing from outside is made (fig. 4.21). As mentioned before, the optimal position

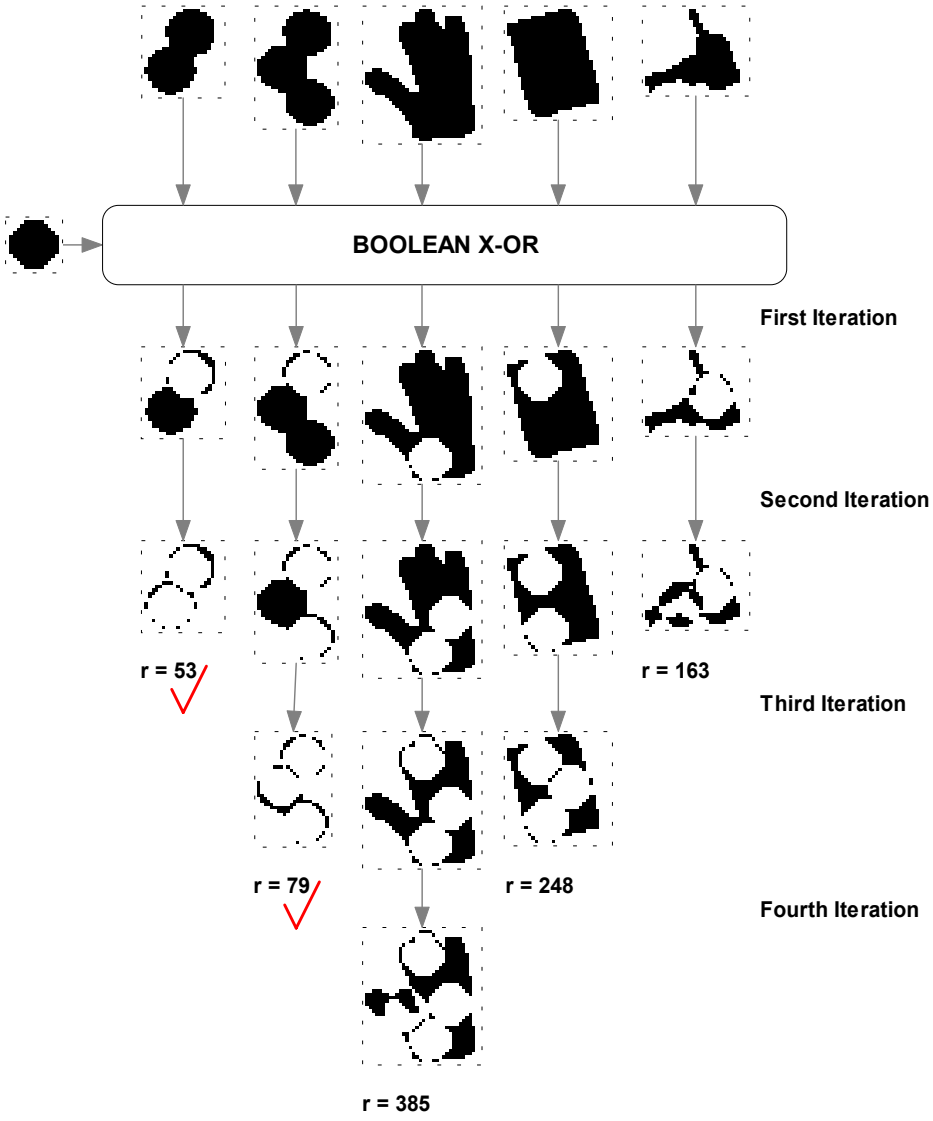


Figure 4.18: Iterative Boolean X-OR computations of various shapes.

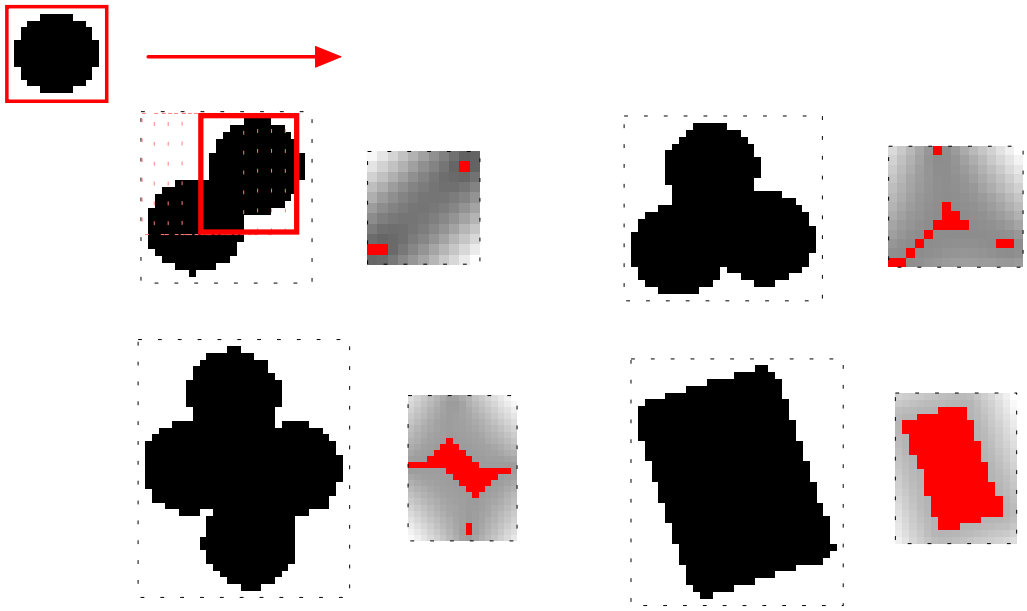


Figure 4.19: Remaining pixel calculation: successive shift of the chipmask over the object, X-OR calculation and summation of the remaining pixel; regions of minimal remaining pixel are marked.

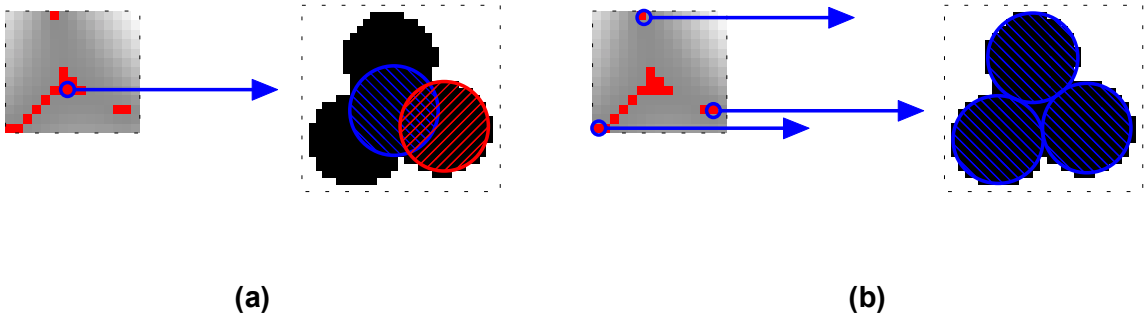


Figure 4.20: Positioning of the chipmask: (a) wrong choosen position, further iterations producing additional remaining pixel; (b) optimal positions, few remaining pixel.

for the whole algorithm may differ from the optimal position for each iteration. Shape deformations caused by low quality acquisition and perspective distortions conduct to the fact, that minima of remaining pixel are found only in the inner part of objects, but the optimal position would be in the outer neighborhood. Again, later iterations would overlap with the current one and more pixel would be added which results in errors. To prevent such situations, the neighborhood of each position is analyzed also and the global amount of remaining pixel is calculated for every distribution of those positions.

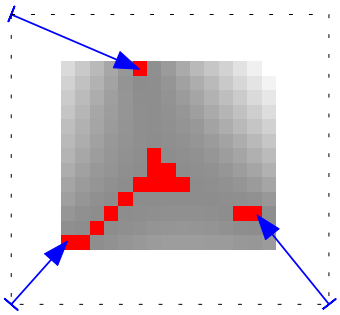


Figure 4.21: Position estimation with region corner distances.

In order to speed up the algorithm runtime, not all positions are used for calculation. The distance from one position to the next is determined also by the amount of the remaining pixel of the prior position. If the amount of remaining pixel is very high in a specific position it is not possible, that a near position will lead to a much smaller amount, so the distance to the next point can be increased, if the amount is very low, each position should be analyzed further and the distance is set to low values as well.

The amount of used iterations for every blob is determined over the following constraints:

- remaining pixelsum < threshold value for chipstacks -> compound chipstacks
- remaining pixelsum < reference pixelsum - tolerance -> too much remains for chipstacks, too less pixel left for another iteration -> no chipstacks
- remaining pixelsum > remaining pixelsum of last iteration -> no chipstacks

The threshold value for detection of chipstacks is dependent from the amount of iterations, it is clear that more compound chipstacks will have more remaining pixel than less compound chipstacks. If the remaining pixelsum is lower than the estimated threshold of the particular iteration, compound chipstacks are detected and the amount of components is equal to the amount of performed iterations. If the sum is higher than this threshold but lower than the reference pixelsum, it is obvious that no chip can be left in the blob and another iteration is not useful, no compound chipstacks are found because of the too high amount of remaining pixel. The truncation condition is set by comparing the remaining

pixelsum with the one of the last iteration, if the amount is increasing it is clear that no more iterations are needed and the algorithm can be halted, no chipstacks are found.

This presented method works well in the prototype environment and proved robust classification results which can be seen in chapter 5.

4.2.5 Card Detection

The detection of cards is the most important vision task. The whole analysis system is using different game states which can only be determined by knowing the card situations on the table. Therefore a robust card detection algorithm is needed to ensure proper functionality. Again the classification of cards is done by comparison of extracted object features with reference card features, which are described in section 4.2.3.

Initialization

The reference features of one single card are determined in an separate initialization step on demand, regarding to be flexible to different locations and card sizes. The segmentation process of the initialization is the same as used for chip initialization. Based on the difference image of the empty table surface and the surface containing one single card the threshold value for binarization is determined and refined in another approximation. Further details are found in section 4.2.4. From the binary image of the single card the reference features are extracted using the same Hough transformation-based algorithm as for consecutive card detection, which will be described in the following section.

Detection

The detection of cards is mainly a single frame image processing chain, but some constraints of former frames are used to simplify the problem set.

For every captured frame the desired region of interest, namely the card region is extracted and binarized using the learned threshold value in the initialization step. Distinct blobs are separated with region labeling, objects which touch the region borders are abolished. These are mainly hands which don't need to be analyzed. Remaining blobs are hole filled, which may appear because of dark card imprints.

Because the dealer cards are the most important ones which need to be found, those blobs which are within the dealer card region are processed first. The algorithm is stopped as soon as one blob is identified as single card or card pile within this dealer card region. If there are no cards inside, the remaining blobs are processed to detect player cards. Some constraints to former frames can be found regarding to the game flow and not all blobs may be processed in the current frame. The following algorithm is simplified to the detection of single cards and piles of two cards which is sufficient for a proper game analysis. The dependencies and conclusions regarding to former frames and the game flow are described in section 4.3.2.

The approach used for chip detection with a binary mask cannot be used to detect cards because they are not rotational invariant. But the fact, that cards are rectangular objects with defined dimensions can be used for their classification, which is independent of their position and orientation.

Pre-classification is done using the pixelsum and compactness (section 4.2.3) of the blobs. Small objects can be separated by a pixelsum threshold value derived from the reference pixelsum of the reference card and a tolerance level. Very compact objects like chips and not compact objects like open hands can be swapped out by compactness range checks. The compactness is calculated with the modified border tracer which is described also in section 4.2.3.

Because the use of compactness alone is not a very robust classifier, not rejected blobs are further processed with a Hough transformation-based strategy. The goal is to get the approximate border lines of the blob and analyze their orientations and intersections, estimate the main object corners and calculate object dimensions. These values can be compared with the reference data and used for proper classification. Details of the Hough transform and its behavior are presented in section 3.2.

The border pixel of not uniquely classified blobs over compactness are transformed into the Hough space (s, θ) using

$$s = x \cos\theta + y \sin\theta. \tag{4.2}$$

Maxima in the Hough space, which represent the main straight lines of the object boundaries need to be found afterwards. Global maxima search in the Hough space accumulator is not possible because the height of the maxima is dependent on the amount of pixel of the corresponding border line and in the neighborhood of maximum values several other high values can be found. So it is possible that values around one maxima are higher than other peaks representing lines, as shown in 4.22 (c). The solution is to search for local maximum values and exclude the neighborhood around the found extrema. This can be done by iterative global maxima search in the whole accumulator, and the neighborhood around a found maxima is cleared. The size of the cleared area can be determined by the shape and size of the desired objects, namely the cards. Figure 4.22 (a) and (b) show the border of a single card and its calculated Hough space (s, θ). Dark regions in the Hough space are high accumulator values, which represent the longest straight lines found in the original image. One single line and its corresponding peak is accentuated, the fact that only the line orientation can be determined from the Hough space, the start- and endpoints need to be calculated separately. For a better view of the accumulator values the Hough space is also shown color coded in three dimensions, the four peaks representing the main border lines which need to be detected can be seen very well.

For proper classification of single cards and piles of two cards up to six independent local maxima are searched within the Hough space. For every distribution of corresponding line pairs the intersection is calculated. The position of each found intersection is checked and those intersections, which are found outside the region borders are rejected. This is done because only those intersections are relevant for the further analysis which are real

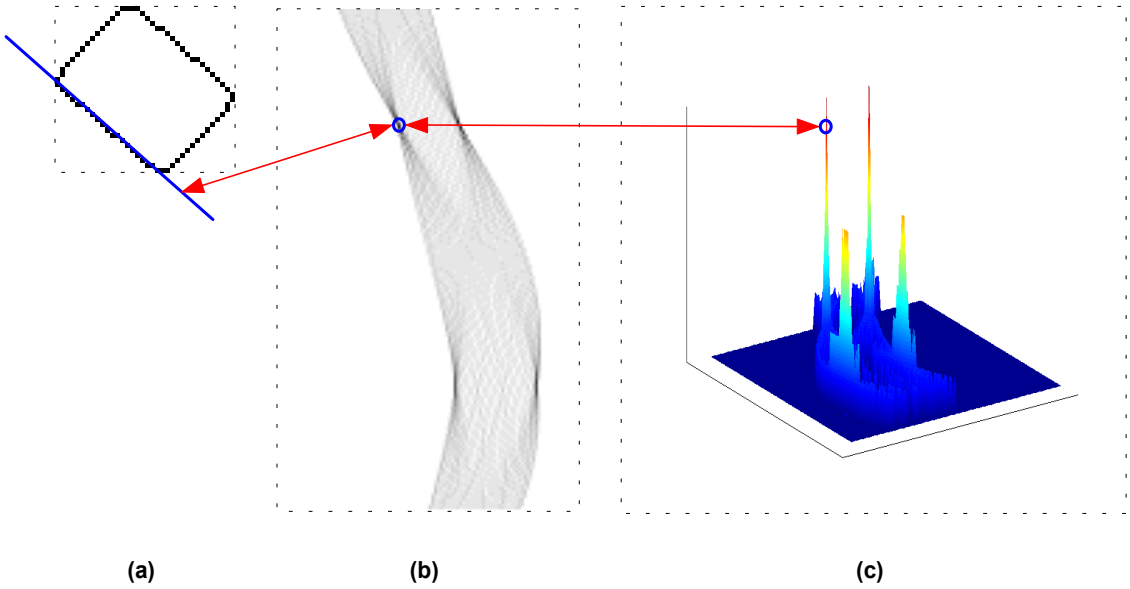


Figure 4.22: Hough Transform of a single card: (a) card boundaries; (b) 2D Hough space (s, θ); (c) color coded 3D Hough space; one straight sample line and corresponding peaks is accentuated.

object corners, points that are not near the object boundaries are intersections of opposite border lines and are not wanted. The idea is to use the amount of resulting corners for a pre-classification. An object with less than four corners cant be a card nor a pile of two cards in any disposition, so they can be classified as non card objects. But two or more inner intersections may lie close to each other and falsify the amount of real object corners, therefore narrow intersections are merged and only the outer one is used as a corner for proper count.

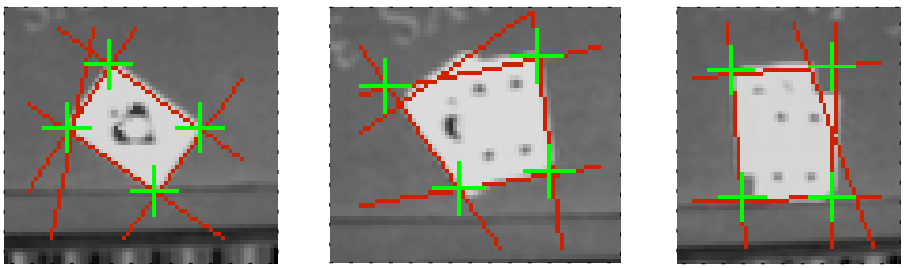


Figure 4.23: Prime straight lines and their intersections of a single card and biles of two cards.

Objects with four or more corners need further analysis. The angles of the prime boundary lines of a single card are 90 degree. Thus, the angles between the found lines are

checked and the amount of normals is determined. To suppress small object distortions the angle is not limited to 90 degrees, a small tolerance range around 90 degrees is used. Objects with less then three estimated normals cannot be a single card or a bile of two cards in any disposition, because at last two corners of a card are visible which lead to three normals. For objects with four or more normals the center is calculated out of the corners and the distances between the center and the corners are compared with the reference distances. If those distances are smaller than the reference dimensions, the object could not be a card and is rejected. Distances within the tolerance interval are classified as single cards, bigger distances leading to card piles.

Classification results using this Hough transformation-based method are shown in chapter 5.

4.3 The Working System

The use of the vision modules solely does not lead to a proper functionality. The knowledge about the Blackjack game and the game-flow itself must be integrated too. For example there is no possibility to distinguish a card from the dealers closed fist in some special cases where the brightness of the hand and the card is almost the same (see fig. 4.25). Another problem caused by a white shirt worn by the dealer can also yield to the detection of a stack of cards. Because of this and similar problems single frame by frame image processing and analysis of the extracted data does not work. Rather an image steam has to be analyzed and the gathered information is appraised by an additional controller. The first section of this chapter describes the streaming facilities of the system. The use of game dependencies for reducing the analysis complexity is presented in the second part. The prototype behavior regarding to the problem set fulfillment is presented with possible additions to the system and at last an overall survey of the implemented prototype including the user interface and interaction possibilities finalizes this chapter.

4.3.1 Image Streaming Context

The image data provided by the acquisition system need to be on a single frame per frame base so that the image based vision modules can be used on the data directly. Therefore the camera is used to supply uncompressed image data which is send to the system on demand. To ensure a continuous Blackjack game analysis without interruption, the running system is requesting permanently images with a defined break time between them. The amount of frames per second is dependent from the acquisition and computer capabilities regarding to the used processing algorithms. The condition for the minimal amount of frames is to detect all changes on the game table raised by the dealer or players. Every chip and card which comes on the table must be visible at least in one frame for initial detection, but to enhance the robustness of the overall system a time window of four frames per second is used. All four frames are processed independent, but the data is merged to one using set for the controller. The advantage of this method is that detection errors in single frames,

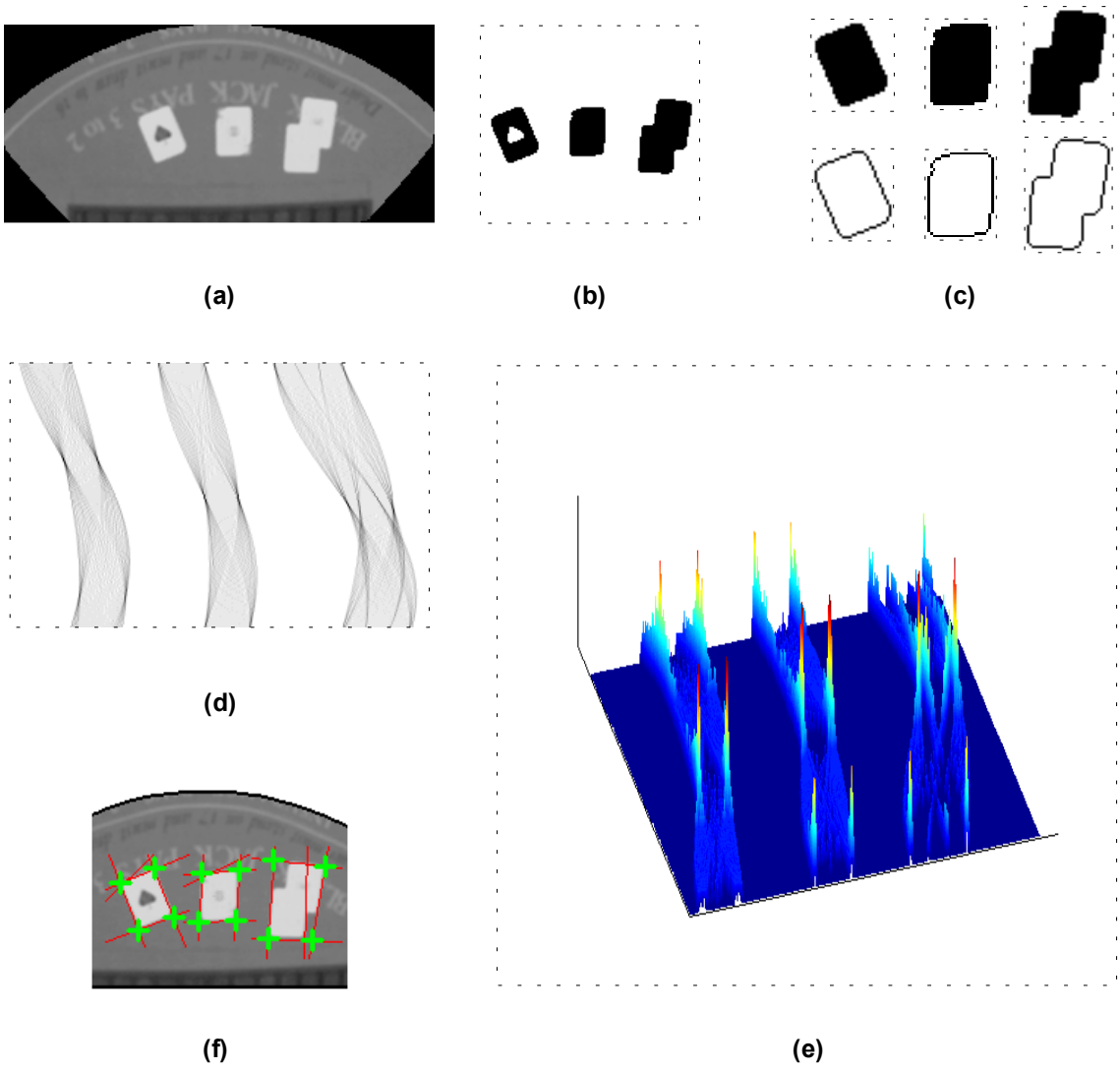


Figure 4.24: Detailed view of card detection steps: (a) cards reagion; (b) binarized; (c) filled blobs and boundaries; (e,f) Hough space; (g) straight lines and resuling intersections.

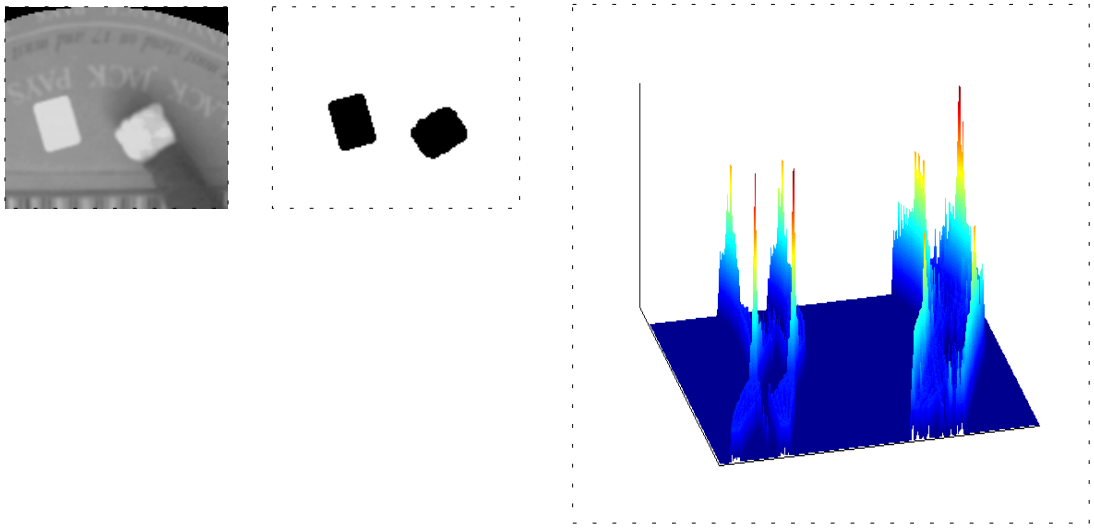


Figure 4.25: Example for identity problems using single frame processing - single card and fist: (a) cards region; (b) binarized; (c) Hough space.

which would lead to overall game-flow analysis errors, are suppressed by the averaging of the time window. One penalty is that changes on the table must stay for more than one frame, because else they get suppressed too. The minimal amount of frames needed for proper data achievement was experimentally determined to three frames per second. With the four used frames per second all occurring actions on the table can be detected, which are all longer visible than one second. The systems follow-up time is lower than half a second in the worst case. This amount of frames is restricted by the used computer and acquisition system, higher frame rates are possible with today's standard computer systems, which may enhance the overall robustness and reduce the follow-up time of the system.

The data which is used for averaging is the count of chipstacks per player box determined by the chip detection algorithm and the card status which is derived from the card detection (see chapter 4.2, section 4.2.4 and 4.2.5). The averaging is done by checking the data of the current frame with the last three ones. If the last three are identical and the current one is changing, the system is still using the data of the recent frame, but the new data is stored for following steps and the oldest one is deleted. Variances in single frames caused by bad image processing or acquisition is not visible to the state controller, real actions on the table will be forwarded to the controller after one frame, when that current one and the last one are identical.

As a consequence of this time window technique moving objects and short occluded cards or chips are not resulting into error-prone evaluation. If cards or chips are concealed by a moving hand for a short time, only one frame will differ. In the next frame the original

data will be found again and the moved hand does not deal to overall errors caused by wrong presumption of vanishing objects. This enhances the robustness of the system in real gaming environment very well. The duration of the maximal possible occlusion time before the system is using the wrong data is dependent of the amount of frames per second and the amount of frames which must be the same to let the changes to the controller. Thus, the behavior can be adapted to desired times.

4.3.2 Logical Game-Flow Controlling

The game-flow dependencies and the usage of different game states (section 2.3) can be used to reduce the complexity of the analysis problem. Furthermore they can be used to simplify the card detection algorithm. The fact that the dealer puts down every card on its own can be used to reduce the card detection algorithm to single card detection. After one card is classified at a specific position on the table it is marked and in the following frames the resulting blob is not further analyzed but it is simply checked if it is still present or not at that noticed position. So when new cards are occluding the initial one, the system does not need to check the blob and therefore no algorithm is needed, which may detect several occluding cards. The justification for this simplification can be found in the fact that the amount of cards in the card region is not needed for statistical analysis nor for game state determination. Only in the dealer card region the case of one single card or more than one cards has to be detected because two cards in the dealer region initiate the *Payment* state. Because the dealer gives himself also one after one card , the detection algorithm need to find the initial card and the following second one, which is partially occluding the first. In following frames, also a marker can be used and the blob need not to be analyzed anymore, as done with blobs in the player card region.

4.3.3 Problem Set Fulfillments and Additions

The implemented prototype meets all of the mandatory requirements and can be used also for further information output. The used methods to fulfill the requirements defined by the problem specification from section 2.2 are described and optional possibilities which may enhance the gathered output of the system are shown as well.

The determination of the defined game states is made solely by the result from the card detection algorithm, which is a defined card status:

- *None* - No cards are detected in the current frame within the card region.
- *Some* - One or more cards are detected within the card region, one single card may also be in the dealer card subregion.
- *More_Dealer* - Two or more cards are detected in the dealer card subregion.

With those card states and their change from frame to frame the game states *Idle*, *Game* and *Payment* can be determined. The initial state is set to *Idle* - the table is empty.

When the dealer starts to put down cards on the table the detection algorithm switches to *Some* and the game state is set to *Game*. The card detection will switch to *More_Dealer* at the time when the dealer gives himself additional cards and the *Payment* state can be set. When all cards are removed from the table and not detected anymore, the card status switches to *None* and the table status is *Idle* again. There was no criterion found which describes the *Shuffle* state in a common way, thus it is not considered at the prototype implementation. An possible dependency on card positions or orientations may be used to determine if it is a game of shuffling, but further problem specification is needed therefore.

The amount of new games can easily counted by the number of *Idle* to *Game* state switches. For the number of participating players the output from the chip detection algorithm is needed in addition. The specific number of found chipstacks in each player region is increased to the overall sum at the upper state switch, thus the initial count of chips represents the participating players because usually every player uses one single chipstack within his player region and other players may put separate chips in region of other players as well. An addition to the prototype which would be very easy to implement is the detection of tip bets. For every player region an additional tip bet region must be defined and with the use of the normal chipstack detection algorithm the tip bet can be determined in the same way as it is done for the amount of participating players at the state switch from *Idle* to *Game*.

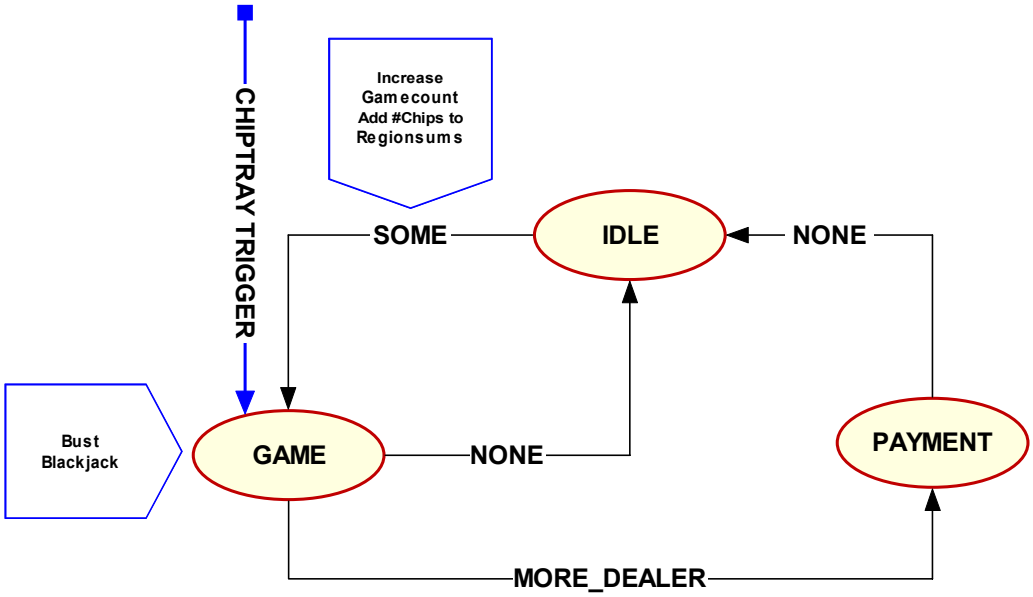


Figure 4.26: Prototype Blackjack flowchart.

Within the *Game* state the system is watching for the player actions. As mentioned before the trigger from the electronic chiptray which represents chip movement is simulated manually. If the trigger of increasing chips in the chiptray is occurring, the system checks the chip detection output with recent frames. The region with less chips in the current

frame must be the one where the *Bust* occurs and is marked. This one is not used in this particular game anymore, because it is allowed for the player to put new chips inside his region for the next game. The same method can be used to detect a *Blackjack*. If the trigger of decreasing chips in the chiptray occurs, the system checks the chip detection output with recent frames and the region with more chips is the one with the *Blackjack*. Also this one is marked and not used anymore in the current game. The timing issues with real chiptray triggers is not taken into account by the prototype, it is clear that the decreasing trigger will occur before the chip detection can find new chips and that time is needed to put down or get chips away from player boxes.

The detection of *Split* and *Double Down* is not implemented in the prototype, but it is also possible with small additional efforts. The chip detection algorithm is used for every frame, thus the increasing amount of chips within a region can be determined. If this happens within the *Game* state and no trigger from the chiptray follows within a small time slot, it must be a *Split* or *Double Down* in that specific region. So the occurrence can be detected, the distinction between them can be made by checking the amount of card blobs within the card region. If the amount increases after the detection a *Split* is made, if the amount keeps the same a *Double Down* occurred.

Another possible addition to the system is the detection of cheating. If the amount of chips decreases within the *Game* state and no increasing trigger from the chiptray follows, the chips must be lost anyhow or vanished in someone's pockets because it is not allowed for the dealer to keep the chips in his hands for use in the following *Payment* state. Several more simple derivations for further information gathering may be found at a deeper look on the system, but that is not the task of this work.

4.3.4 User Interaction - The Prototype Interface

The implemented prototype consists of the game analysis algorithmic, the image acquisition and processing modules and an easy to use graphical user interface. This interface is added for better user interaction and visualization of the system output. This part of the prototype is separate from the algorithmic code and can be changed with low effort for a final implementation where no normal personal computer including a monitor display is used. It is the only part of the code beside the used acquisition drivers which is platform dependent because of the use of the Microsoft's graphical interface, the Microsoft Foundation Classes [Mic01]. The rest of the code is kept platform independent and can be ported to any other computer platform and operating system.

The advantage beside user friendly communication is the feature that every step in the image processing chain can be visualized and controlled. Additional experiments and new code can be verified easily. The use of the graphical interface and the visualization of several images reduces the overall performance of the analysis system, but even with the use of it the requirements on the system are met, thus without the interface more capturing frames per second would be possible to check.

The interface is split into four different windows:

- main interaction window
- capturing window
- analysis output window
- image viewer window.

The main interaction window is the one where different tasks can be started or stopped, events are triggered and options are set using menu items. Optional runtime and debugging informations can be visualized, for example elapsing times for different algorithm steps or extracted object features for classification. All possible user interactions (see fig. 4.27) are described briefly and references to the specific section with details are given. Menu push-buttons are represented by (P) and option toggles by (T).

- Main - Major program menu.
  - Device - Acquisition device settings menu.
    - Video Source Dialog (P) - Standard video source popup dialog (Video for Windows) with the possibility to change camera settings.
    - Video Format Dialog (P)- Standard video format popup dialog (Video for Windows) for changing the camera image format.
    - Freeze Image (T) - Hold on framegrabbing and use the current visible frame until the grab is continued by another toggle of this option. This is very useful for checking image processing behavior on static scenes.
  - Settings - Program settings menu.
    - Load (P)- Former saved initialization parameters can be loaded again.
    - Save (P)- Learned initialization parameters are saved.
  - Save Frame (P) - The current visible captured image frame is saved in a image file.
  - Recording (P) - The captured image stream is saved in a motion file.
  - Exit (P) - Program termination.
- Initialization - Needed initialization tasks which are needed prior the analysis.
  - Empty Scene (P) - The current captured frame is used as empty reference scene for later difference image calculations (see sections 4.2.1, 4.2.4 and 4.2.5).
  - Card Region (P) - The current captured frame is used for calculating the card region, which is masked by a bright sheet of paper on the table (section 4.2.1).
  - Dealer Card Region (P) - The current captured frame is used for estimation of the dealer card subregion, also masked by a bright sheet of paper on the table.

  - Chip Regions (P) - The current captured frame is used for determination of all player regions, which are masked by separate sheets.
  - Learn 1 Chip (P) - The current captured frame is used for reference chip feature extraction, the chip must be placed within the card region and should be placed on a uniform and center part of the table surface (section 4.2.4).
  - Learn 1 Card (P) - The current captured frame is used for reference card feature extraction (section 4.2.5).
  - Reset Statistics (P) - All extracted analysis results are resetted.
  - Use Card Markers (T) - Determines if card markers are used after initial card detection or not. Switching this option off is useful for some card detection experiments, for proper overall analysis this option must be active.

- Output - Optional informations shown in the main interaction window.

  - Elapsed Times (P) - Elapsed times of current algorithm steps.
  - Chip Features (P) - Extracted features from the current analyzing objects found in the player regions from the current captured image (section 4.2.3).
  - Card Features (P) - Extracted features from the current analyzing objects found in the card and dealer card subregion (section 4.2.3).
  - Region: Cards (T) - Show the card region border in the current output image in the image viewer window.
  - Region: Dealer (T) - Show the dealer card subregion border.
  - Region: Chips (T) - Show the player region borders.
  - Captured Data (T) - Show the captured data in the current output image. Switching off will show only markers of classified objects, found lines and active region borders, else the captured image data is also added to the output image in the viewer window.

- Toggle Timer (T) - The detection algorithmic is triggered by a defined timer event depending on the used frame rate. Switching off the timer will halt the overall analysis process and static images can be visually checked.

- Chip Tray + (P) - Simulates the incoming chips trigger event from the electronic chiptray.

- Chip Tray - (P) - Simulates the outgoing chips trigger event from the electronic chiptray.

The capturing window shows the image data grabbed by the acquisition system. The whole computed analysis output from the system is shown in the analysis output window.

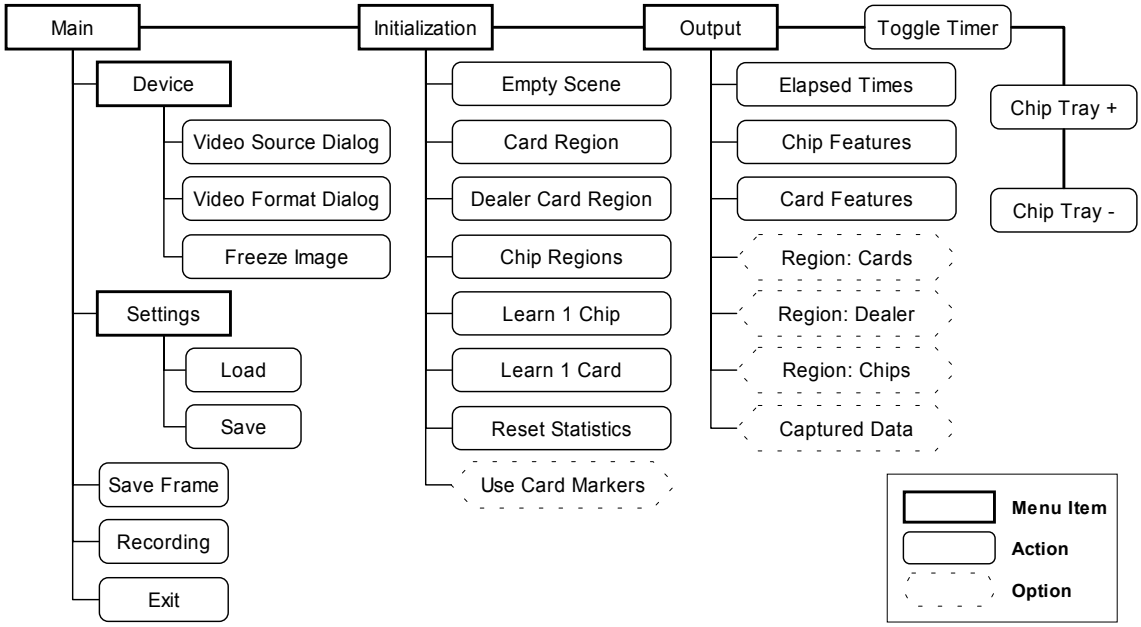


Figure 4.27: Main interaction window menu chart.

The image viewer window is optional and can be used to look deeper at specific image processing steps for human control of the algorithms. Figure 4.28 and 4.29 shows all four windows of a running prototype session.

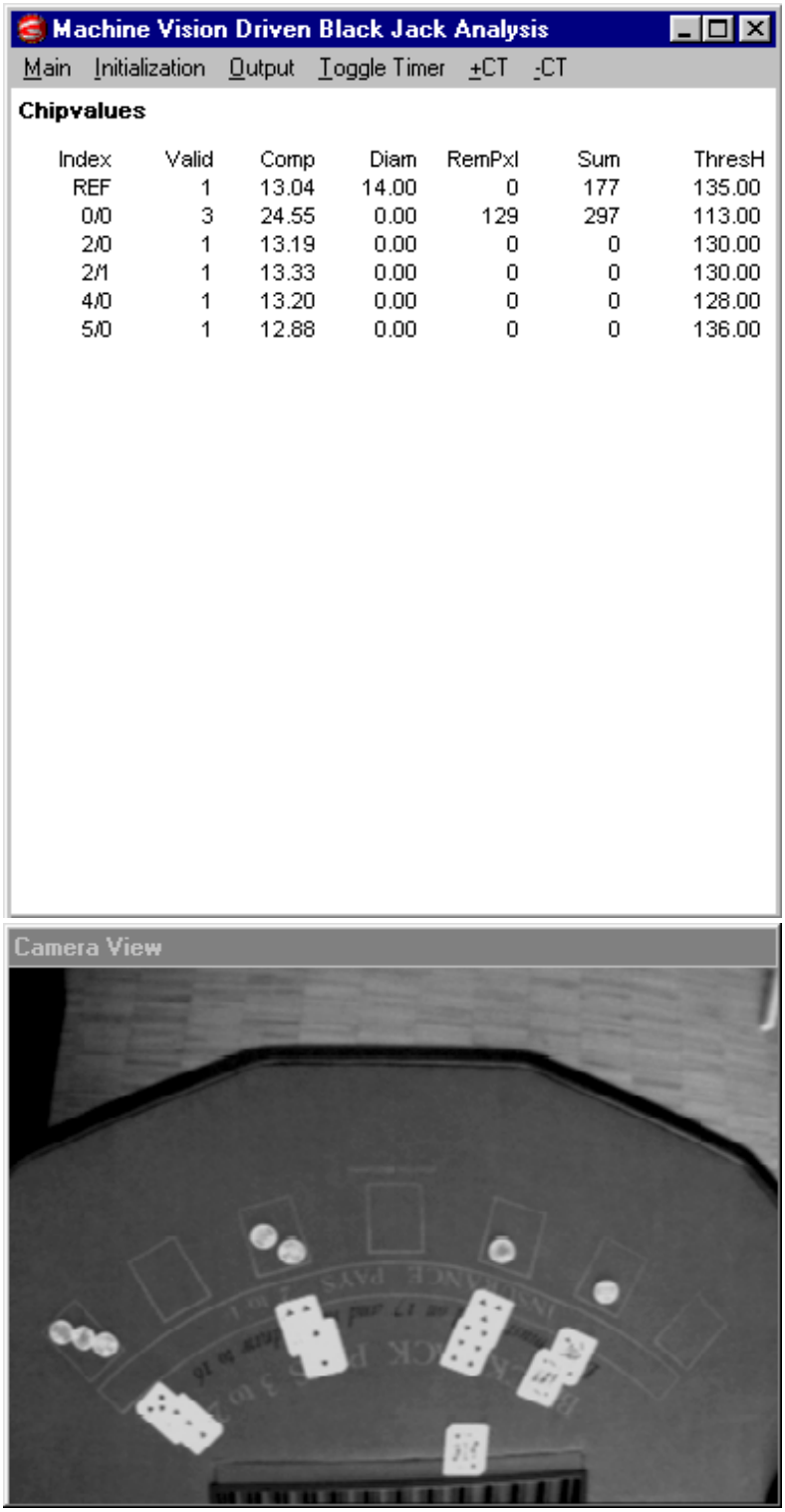


Figure 4.28: Prototype interface: Main interaction and capturing window.

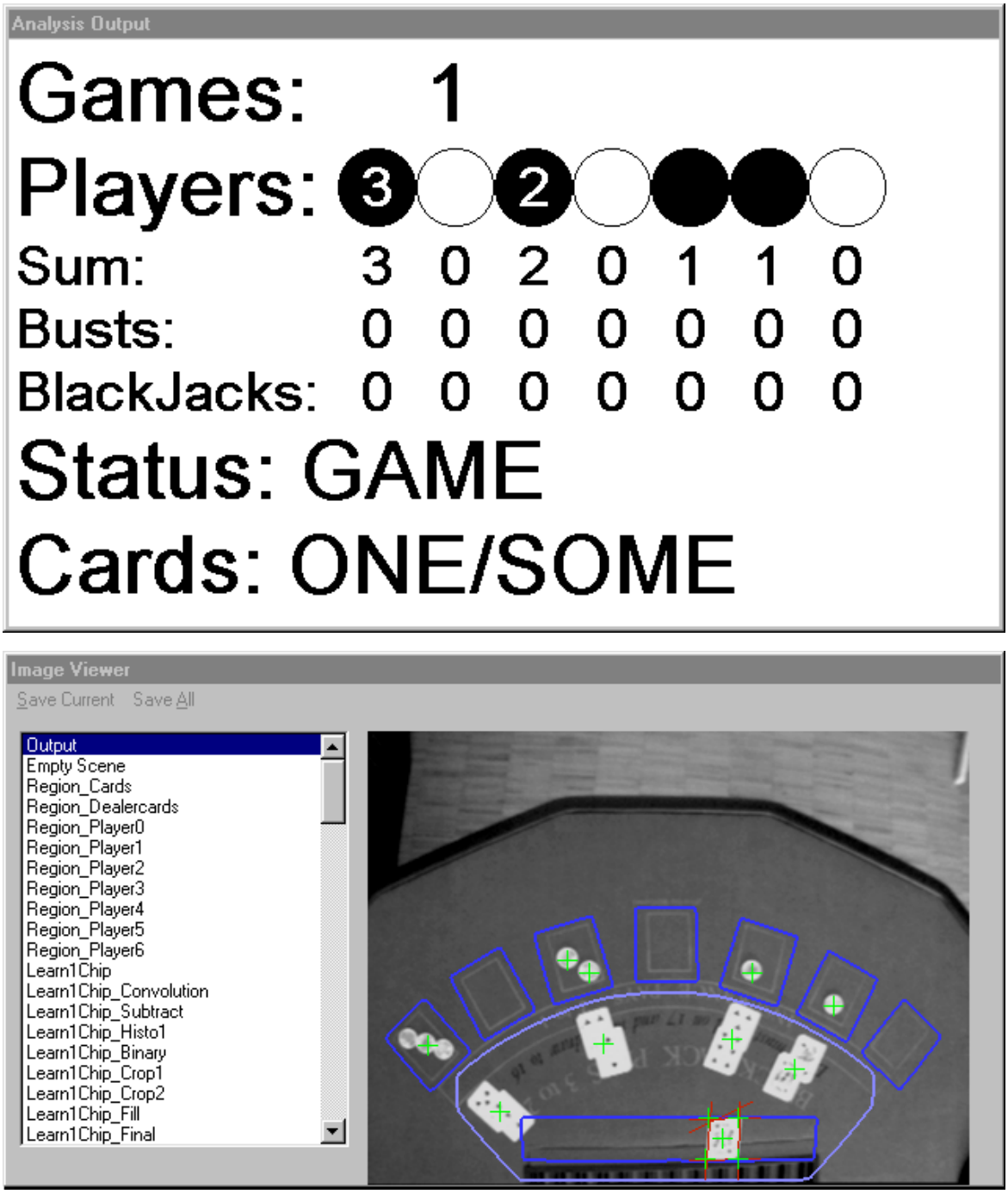


Figure 4.29: Prototype Interface: analysis output and image viewer window.

Chapter 5

Experiments

Initial experiments on which this thesis is based were performed in a feasibility study [SZ98], which was started in October 1998. Regarding to those experiments the prototype environment was built and further alternative algorithms were added and tested. The initial evaluations of various image processing algorithms are described briefly with some derived perceptions in the beginning of this chapter. The prototype experiments are described in more detail in respect of stand-alone chip and card detection and real game-play analysis under various circumstances.

5.1 Feasibility Study

The main goal of this initiated study was to determine if image processing could be used for Blackjack game analysis. Therefore a sample Blackjack game was recorded with a camera. The resulting video was used for the initial experiments, sample frames of it can be seen in figure 5.1.

Figure 5.1: Sample frames from the Blackjack video.

On such sample frames, which were chosen manually out of the video, several image processing algorithms were tested. The purpose was chip-stack and card detection for further analysis. Chip detection was tried using template matching with normalized cross correlation [MC95] at first. The experiments showed that correlation does not provide

useful results for chip detection because of the non uniform chip surfaces causing rotational variances. Regarding to runtime minimization binary image processing was the main direction for following attempts. Threshold values were determined manually and morphological operations like erosion, dilation and opening (section 3.1) were analyzed for image enhancement which brought good initial results. A novel approach for chip detection was the use of Boolean X-OR computations, the improved version of that algorithm is also used in the prototype (section 4.2.4). Cards were detected by simple high-valued thresholding and pixel counting, which exposed as a wrong and non-robust technique in later experiments. Difference images from game scene frames with empty table surface frames were used for separation of the static table surface and its sensitivity to vibrations was taken into account. The general use of difference images for subtraction of static table surface overprints was rejected because of object fragmentation caused by the subtraction and derived object recognition errors.

Most of those algorithms are improved and used also in the prototype and are described in prior chapters, thus only the resulting cognitions from the feasibility study are presented here:

- Blackjack analysis is possible using image processing
- the complexity of needed algorithms permits a real-time application
- analysis robustness can be increased with restrictions like regions of interest and the use of the game-flow.

Those initial results and perceptions leaded to the construction of the analysis system prototype. The experiments using this prototype environment are presented in the following section.

5.2 Results of the Prototype

The experiments using the prototype environment are divided into stand-alone chipstack detection, detection of one and two cards and usual game-play analysis. Initial tests are used to estimate proper object features and tolerance levels for the object classification, further experiments refine some parameters and show the behavior under different circumstances, for example with added noise or blurred input data.

5.2.1 Stand-alone Chip Detection

Stand-alone chip detection is started with single chip detection. Different algorithms and object features for classification are tested with different tolerance levels to estimate the classification quality. The used test objects can be seen in figure 5.2: different chips and other rectangular obstacles like a card are used for evaluation. The initial features for classification are compactness, object pixelcount and diameter. The next experiment

set shows the presented Boolean X-OR computations and classification results using the remaining pixelsum. A standard Hough transform technique is evaluated as well. At last a combination of compactness and X-OR computation is tested for single chip detection.

Compound chipstack detection is tested by the Hough transform technique and the combined X-OR algorithm. Several compound dispositions of two, three and four chips from the test data set are evaluated. The final experiment for stand-alone chip detection is done with the proposed combined X-OR algorithm with one set of equal printed chips under changing light environment.

Figure 5.2: Test data set for chip detection: various chips, a card and other rectangular obstacles.

The initial test using the learned compactness of the reference chip with allowed tolerance of one percent used for chip classification resulted in noise dependent unsteadiness. None of the chips were classified robust in more frames, only the rectangular objects were properly classified because of their much lower compactness. Better results are achieved using larger compact tolerance levels, which is shown in table 5.1. Each table column illustrates the results for one specific object (chip or rectangular object), the numbers indicate the amount of correct or wrong classified ones. Every object from the test data set is successively placed in all of the seven player boxes and watched for several frames. The quoted percentages are calculated by summation of all correct classified objects in one group (chips or rectangular objects) divided by the total amount of detection attempts. The amount of proper classified chips increases with raising the allowed compactness tolerance, but more failures occur at other objects (the rectangular objects of the test data set). The main problem for the chip errors is the not perfect round shape and perspective distortions, which lead to compactness variations, which get even larger at the used low image resolution. Another problem is, that dark chips may pry and the compactness changes drastically (for example chip #7).

The second experiment set (table 5.2) illustrates the results of the presented X-OR computation (see section 4.2.4) using different tolerance levels for the remaining pixelsum. The value is calculated by the denoted percentage from the learned pixelsum of the reference

Compactness	Chips										%	Others					%
±5% Ok	7	1	5	6	7	7	1	0	7	7	69	7	7	7	7	7	100
Not Ok	0	6	2	1	0	0	6	7	0	0		0	0	0	0	0	0
±10% Ok	7	5	7	7	7	7	3	0	7	7	81	7	7	7	7	6	97
Not Ok	0	2	0	0	0	0	4	7	0	0		0	0	0	0	1	3
±15% Ok	7	7	7	7	7	7	5	1	7	7	89	7	7	7	6	0	77
Not Ok	0	0	0	0	0	0	2	6	0	0		0	0	0	1	7	23

Table 5.1: Single chip classification using compactness with different tolerance levels in respect of the reference compactness.

Remaining Pixelsum	Chips										%	Others					%
20% Ok	5	5	6	6	7	5	6	1	4	7	74	7	7	7	7	7	100
Not Ok	2	2	1	1	0	2	1	6	3	0		0	0	0	0	0	
25% Ok	7	6	7	7	7	7	7	2	7	7	91	7	7	7	7	7	100
Not Ok	0	1	0	0	0	0	0	5	0	0		0	0	0	0	0	
30% Ok	7	7	7	7	7	7	7	5	7	7	97	7	7	7	7	6	97
Not Ok	0	0	0	0	0	0	0	2	0	0		0	0	0	0	1	
35% Ok	7	7	7	7	7	7	7	7	7	7	100	7	7	7	6	4	86
Not Ok	0	0	0	0	0	0	0	0	0	0		0	0	0	1	3	

Table 5.2: Single chip classification using the remaining pixelsum from Boolean X-OR computations regarding to the pixelsum of the reference chip.

chip. It can be seen that the classification using the Boolean X-OR works better than the one using compactness, the quality of the classification also depends on the shape of the chips. If chips are distorted or not circular in the binary image, which may caused by too different chip colors, this technique can't work properly, because the amount of remaining pixel would depend on the rotational pose of the chip (for example chip #8 is not very round in the binary image which is caused by it's color and the used grey-level thresholding).

A Hough transform based algorithm (see section 3.2.2) is used in comparison to the X-OR technique, which tries to extract circles from the binary input image. The used classification feature is the maximum value in the calculated Houghspace accumulator using different percentages of the amount of border pixel from the reference chip. Table 5.3 shows that the results are not as good as the ones using the X-OR algorithm. The main problems are small position dependent projective distortions and the low resolution of the input images. The implemented Hough algorithm is using the learned chip diameter to build the Houghspace, which can be made two-dimensional in this special case. Better results can be achieved using a three dimensional Houghspace varying the diameter, but that is not possible in this field of work because of the real-time issue. But even then, the resolution is still very low, circles and rectangles have very much common border pixel which make the classification using the Houghspace accumulator maxima error-prone.

Final single chip detection experiments are combined in table 5.4. The classification is

Houghspace Max.	**Chips**										**%**	**Others**					**%**
80% Ok	6	1	6	5	7	5	5	0	7	6	69	7	7	7	7	7	100
Not Ok	1	6	1	2	0	2	2	7	0	1		0	0	0	0	0	
70% Ok	6	1	6	6	6	7	5	0	7	7	73	7	7	5	7	6	91
Not Ok	1	6	1	1	1	0	2	7	0	0		0	0	2	0	1	
60% Ok	7	5	7	7	7	6	7	2	7	7	89	7	7	1	6	2	66
Not Ok	0	2	0	0	0	1	0	5	0	0		0	0	6	1	5	

Table 5.3: Single chip classification using accumulator maxima from the Hough transformation in respect of the amount of border pixel from the reference chip.

	Chips										**%**	**Others**					**%**
Ok	7	7	7	7	7	7	7	6	7	7	99	7	7	7	7	7	100
Not Ok	0	0	0	0	0	0	0	1	0	0		0	0	0	0	0	
10% S&P Noise Ok	7	7	7	7	7	7	6	3	7	7	93	7	7	7	7	6	97
Not Ok	0	0	0	0	0	0	1	4	0	0		0	0	0	0	1	
20% S&P Noise Ok	6	6	6	6	5	5	1	1	6	6	69	7	7	6	7	6	94
Not Ok	1	1	1	1	2	2	6	6	1	1		0	0	1	0	1	
Gauss $(5x5, \sigma = 2)$ Ok	7	7	7	7	7	7	6	6	7	7	97	7	7	6	7	7	97
Not Ok	0	0	0	0	0	0	1	1	0	0		0	0	1	0	0	
Gauss $(3x3, \sigma = 1)$ Ok	7	7	7	7	7	7	7	7	7	7	100	7	7	7	7	7	100
Not Ok	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	

Table 5.4: Single chip classification using the proposed algorithm with remaining pixelsum from the X-OR computations and compactness in common; influence by Salt&Pepper noise and Gaussian blurring is evalutated.

done by compactness ($\pm 10\%$) and Boolean X-OR (30% remaining pixel) in common, which achieved the best results. Various circumstances are tested: Salt&Pepper noise was added to the input images to evaluate the behavior of the image processing and classification algorithm with noisy image acquisition or data transmission. At higher noise levels darker chips lean towards to force open, which results in too much remaining pixel, because the hole filling does not work properly. On the other hand the input images are Gaussian blurred (see section 3.4) to simulate bad camera focus adjustment. It can be seen, that small blurring may enhance the classification quality because distortions caused by the perspective or non uniform chips are decreased.

The achieved results from single chip detection and estimated tolerance levels are used now for initial compound chipstack detection. Because of the occurring widespread compactness interval of compound chipstacks, this feature is not possible for classification. The Hough-based strategy showed additional errors because the amount of border pixel per chip decreases because of compound components and the tolerance level must be increased. Thus, even more classification errors occur because of similar border shapes of

Houghspace Max.	**2 Chips**					%	**3 Chips**					%	**4 Chips**					%
60% Ok	6	7	7	6	7	94	5	5	4	5	6	71	3	3	0	6	2	40
Not Ok	1	0	0	1	0		2	2	3	2	1		4	4	7	1	5	

Table 5.5: Compound chipstack classification using accumulator maxima from the Hough transformation.

	2 Chips					%	**3 Chips**					%
Ok	7	7	7	6	7	97	7	7	6	6	7	94
Not Ok	0	0	0	1	0		0	0	1	1	0	
Gauss ($3x3, \sigma = 1$) Ok	7	6	7	7	7	97	7	7	6	7	7	97
Not Ok	0	1	0	0	0		0	0	1	0	0	
	4 Chips					%	**Others**					%
Ok	7	7	7	6	5	91	7	7	7	7	7	100
Not Ok	0	0	0	1	2		0	0	0	0	0	
Gauss ($3x3, \sigma = 1$) Ok	7	7	6	7	6	94	7	7	6	7	7	97
Not Ok	0	0	1	0	1		0	0	1	0	0	

Table 5.6: Compound chipstack classification using the remaining pixelsum of Boolean X-OR computations.

compound chipstacks and other rectangular objects.

Table 5.4 shows results for two, three and four compound chips of the test data set using 60% of the amount of border pixel from the single reference chip as Houghspace accumulator threshold for classification. The main disadvantage of the Hough-based method is the fact, that the classification threshold of the Houghspace accumulator maxima must be set to a single value. It is not possible to use different tolerance levels for classification of two, three and more compound chips. But this adaptive behavior can be provided using the Boolean X-OR method in an iterative approach. The tolerance levels can be changed in each iteration step and thus the amount of allowed remaining pixel can be increased with the amount of compound chips. This adaptation leads to much better classification results, which can be seen in table 5.7. Gaussian blurring ($3x3$ kernel, $\sigma = 1$) enhances the classification results for compound chips, but other objects may get rounded and wrong classified. Regarding to this results, the combined X-OR method with compactness classification for single chips is the proposed algorithm for chip detection.

The final experiment for stand-alone chip detection is using the proposed X-OR based detection algorithm,but instead of the test data set a heap of equal printed chips with different colors are used. This should simulate a typical casino environment, where no different types of chips are used on one table. All player boxes are filled randomly with one to four chip-stacks, then one card is placed on the table to initialize a game for chip-counting. The results (see table 5.7) are controlled afterwards manually from the captured movie. To simulate changing light environment, additional light sources were used with

random position and intensity. The initialization step to learn the reference chip features and the chip-mask is done between different runs with different colored chips.

Compound Chip-stacks	Samples	Errors	Accuracy
1	100	0	100 %
2	100	1	99 %
3	100	2	98 %
4	100	2	98 %

Table 5.7: Final results for stand-alone chip detection using the proposed iterative X-OR based method and compactness classification in common.

The implemented algorithm using iterative Boolean X-OR calculations and compactness classification in common proved robust behavior. Adaptive tolerance level adjustment improves the classification results in comparison to the Hough based method. Few detection errors occurred because of too much different colored chip surfaces and their reflection which leads to slightly different sizes of the binarized chip or by too capacious distortions caused by the acquisition perspective. This error increases with the number of compound chip-stacks, but is understated by the fact, that more than two compound chip-stacks occur only seldom in practice. Due to this experimental results, the Boolean X-OR based algorithm with remaining pixel classification in combination with compactness classification for one chip is the proposed method for chip detection.

5.2.2 Stand-alone Card Detection

Stand-alone card detection is started with single card detection using compactness and object pixelsum. Further experiments are evaluating the Hough based algorithm, which uses the extracted information about lines and their intersections for additional classification. The used test objects can be seen in figure 5.3: different cards and smaller parts, compound chips and some hand patterns. Some of the cards are manipulated with a dark marker on the edges and on the side to simulate possible segmentation errors caused by darker parts of cards or segmentation errors causing shape differences of the cards. The hand patterns should emulate the behavior of the card classification, when original hands are that bright, that they are segmented to objects and must be analyzed by the algorithm. Note, that this is not a common case, usually hands are too dark and moving on the table. Thus, they are not regarded by the system because of the used frame window technique (see section 4.3.1). Nevertheless it should be possible to separate the hands from the cards using the card classifier.

The initial experiments showed, that pixelsum and compactness is only limited useful for classification. If card borders break in caused by dark imprints, the compactness varies very much, and too high tolerance levels are needed, which result in error-prone hand classification. Better results were achieved using the Hough transformation to detect main lines from the object borders and analyze their intersections, distances and angles between

Figure 5.3: Test data set for card detection: various manipulated cards and smaller parts, hand patterns and compound chips.

them. Up to six Hough maxima were extracted from the Houghspace accumulator and all possible intersections were calculated and further processed. The number of found normals and parallels are used for classification with additional distance checks between the regarding intersections. The minimal used value d_{Min} of the Houghspace maxima which must be extracted, was determined experimentally using the card disposition shown in figure 5.4. Lines with less pixel are not needed for the classification, thus all Houghspace peaks lower than that value must not be extracted.

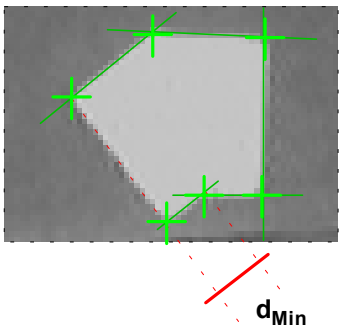


Figure 5.4: Minimal needed length for line detection.

Table 5.8 shows initial results using compactness and pixelsum in common with the informations from the extracted lines by the Hough transformation. Cards and other objects are thrown randomly on the table and the classification results are checked. For evaluating the detection of two cards another card is thrown on the table in such a way that the cards are occluded partially. Most errors results from partially breakup of card borders leading to too large compactness changes (up to three times the original compactness of one card). A problem was also the proper classification of four chips combined to a rectangular compound stack, which has almost the same size as one card. After using the test objects, a normal non-manipulated card pile is used and from those results some classification parameters are refined: the angle tolerance was increased to 12 degree and

	1 Card	%	**2 Cards**	%	**Others**	%
Test Objects Ok	24	96	22	88	13	93
Not Ok	1		3		1	
Non-manipulated Cards Ok	25	100	21	84		
Not Ok	0		4			
Tolerance Refinement Ok	25	100	24	96		
Not Ok	0		1			

Table 5.8: Stand-alone card detection using compactness, pixelsum, line and intersection analysis

	1 Card	%	**2 Cards**	%
Gauss $(3x3, \sigma = 1)$ Ok	25	100	24	96
Not Ok	0		1	
10% S&P Noise Ok	25	100	25	100
Not Ok	0		0	
20% S&P Noise Ok	25	100	24	96
Not Ok	0		1	
30% S&P Noise Ok	25	100	16	64
Not Ok	0		9	

Table 5.9: Stand-alone card detection under various circumstances: Salt&Pepper noise and Gaussian blurring.

the compactness interval was reduced, because the border of normal cards didn't trend to breakup.

The implemented classification algorithm with the estimated tolerance levels was tested also under various circumstances. Table 5.9 shows results from experiments using Gaussian blurring and additional noise generation, the algorithm works very well with blurred and noisy data, thus it is the proposed method for card detection.

Final stand-alone card detection was done using the proposed method for detection of typical card dispositions. First one, than another partially occluding card is put down on the table into the dealer card subregion. The amount of started games by the first card and payment phases caused by the second card is controlled afterwards.

Additional experiments were made with optional distance transformation [Bor86] of the object borders. The goal was the separation of those intersections, which result from lines

Cards	**Samples**	**Errors**	**Accuracy**
1	100	0	100 %
2	100	1	99 %

Table 5.10: Results for stand-alone card detection using typical card dispositions.

of different cards representing no real corners. Because of error-prone behavior and timing issues using the slow distance transform, this approach was rejected.

Card detection using compactness and pixelsum does not lead to proper classification. The Hough based method which extracts the main border lines and uses informations about those lines and their intersections provided robust results and is the proposed algorithm for card detection.

5.2.3 Game-play Analysis

The behavior of the prototype at usual game-play is done by recording several runs of games with more or less different players. The output from the analysis is compared afterwards manually with the recorded video. The proposed algorithms for chip and card detection are used with the estimated tolerance levels from former stand-alone detection experiments.

As it can seen from the game-play results (see table 5.11) the behavior of the prototype is highly accurate and fulfills the requirements. The error at the third run occurred in consequence of too long card occlusion by a hand which leads to a too high game count, at the fourth run a chip-stack was wrong classified.

Run	Games	Chipstacks	Busts	Blackjacks	Overall Err.	Tot. Accuracy
1	7	48	8	2	0	100 %
2	7	9	2	0	0	100 %
3	21	43	8	3	1	95.24 %
4	11	40	11	3	1	97.50 %
5	7	43	12	4	0	100 %
	53	183	41	12	2	97.58 %

Table 5.11: Statistical game-play analysis using the proposed detection algorithms fused with logical game-flow informations.

	Method	Avg. Runtime [ms]
Single Chip Detection	Compactness & Diameter	25
	Boolean X-OR	26
	Hough transformation	36
	Compactness & X-OR	28
Chipstack Detection	Hough transformation 2/3/4 Chips	30/35/37
	Boolean X-OR 2/3/4 Chips	40/51/68
Card Detection	Compactness & Hough 1/2 Cards	22/35

Table 5.12: Average runtimes of evaluated algorithms.

To give an overview of the complexity of the used algorithms, table 5.12 shows average runtimes of the implemented algorithms. The runtimes are measured using the chip

detection algorithms on 15 independent objects (single chips or compound chipstacks) on the table, the times for card detection are for one single card and two partially occluding cards. It can be seen that this implemented methods work very fast and thus fulfill the required real-time capabilities of the whole analysis system.

Summarizing the results from the above experiments, the implemented image processing algorithms for chip and card detection proved robust and accurate behavior. In the case of chip detection, the adaptive and iterative X-OR based method provides better results then standard techniques like the Hough transformation for circle extraction. For card detection, the additional Hough transformation step and analysis of extracted border lines and their intersections admit robust classification which is very insensitive to noise.

The overall game-play can be analyzed very robust by fusing the results from the image processing detection algorithms with logical game-flow. The complexity of the implemented algorithms allow real-time usage of the proposed system.

Chapter 6

Conclusion and Outlook

6.1 Summary

In this thesis a novel machine vision driven real-time Blackjack analysis system has been presented. By analyzing the various states of the game and the use of non-visual information as chip flow in the chip-tray the requirements of the machine vision part can be greatly reduced. Robust algorithms for the remaining two tasks, namely chip and card detection, have been developed and evaluated. All algorithms have been kept fast and simple in order to assure real-time application.

Fast adaptive gray-level thresholding and region labeling techniques were used for segmentation, simple object features like compactness, diameter, width and height are extracted from the separated binary object blobs. The object classification is done by comparison of the extracted features with reference features. In the case of chip detection the proposed combination of classification by compactness for one chip and the amount of remaining pixel from iterative X-OR computations with the reference chipmask for one chip and compound chipstacks brought very accurate and robust results. Single chips and compound chipstacks are detected with more than 98% accuracy at usual game environment. Card detection by the suggested Hough transformation based algorithm and classification by informations about the extracted lines and their intersections worked also robust. Cards are detected with more than 99% accuracy.

To be adaptive to various game table surfaces and different chip and card types, initialization processes were made to define the regions of interest and learn the desired objects, namely cards and chips, and calculate their reference features on demand.

By not only using single frame image processing algorithms but fusing the results with logical game flow informations and using a streaming context the system has met the mandatory and important requirements:

- the table states *Idle*, *Game*, *Payment* are proper estimated,
- game informations like the amount of games and participating players are counted,
- and detection of player events like *Bust* and *Blackjack* are detected.

Experiments on the simplified laboratory prototype have demonstrated the feasibility of the presented approach and proved the robust behavior of the system. The total accuracy of the whole game-play analysis is more than 97%.

6.2 Future Work

Although the mandatory aspects have already been implemented there are some key features that remain to be done in the future.

First of all the target computer hardware and peripherals need to be specified, the platform and the type of user interaction must be defined as well. The prototype interface need to be adapted and the image acquisition system updated for the final real environment implementation. The communication with an operating chip-tray need to be implemented in order to consider the chip-tray triggers and the proper timing. The data communication with a central server station which stores the analysis results for further statistical calculations must be specified and implemented too.

Depending on the desired position and quality of the capturing device, camera calibration and image deskewing must be done in order to reduce projective distortions which may cause too big differences in chip and card sizes depending on their position on the table making the implemented algorithms useless.

The amount of used frames and the size of the time window used by the controller need to be adapted to the real environment leading to the desired system response times and behavior. A compromise between the systems follow-up time and the longest possible time of object occlusion need to be specified as well.

The prototype is using the GIPLIB for several algorithms and data containers. Because of the open and modular object oriented design of the platform independent library it is clear that direct C code would be more efficient, but less readable and expandable. If the runtime of the vision modules must be cut down, those need to be further optimized and direct C code without the usage of the GIPLIB should be achieved.

Bibliography

[Bal81] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981.

[BBR+99] G. Bachler, M. Berger, R. Röhrer, S. Scherer, and A. Pinz. A Vision Driven Automatic Assembly Unit. In *Proceedings of the 8th Conference on Computer Analysis of Images and Patterns*, pages 375–382, Ljubljana, September 1999.

[Boe01] Boeder. www.boeder.com, 2001.

[Bor86] Gunilla Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

[BPPP98] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Active object recognition in parametric eigenspace, 1998.

[Dan98] Daniel Rutter. hwww.dansdata.com/quickcam.htm, 1998.

[DH72] R.O. Duda and P.E. Hart. Using the Hough transform to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[Don00] Donrey Media Group. CasinoGaming.com. www.casinogaming.com, 2000.

[Gip98] GIPLIB - General Image Processing Library. Technical report, Institute for Computer Graphics and Vision, Graz University of Technology, 1998.

[GRI99] GRIPS Electronic GmbH. www.grips.com, 1999.

[GW93] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison Wesley, 1993.

[Hou62] P.V.C. Hough. A method and means for recocognizing complex patterns. US Patent 3.069.654, 1962.

[HS92] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision*, volume 1. Addison-Wesley, 1992.

[Ins01] Institut für Machinelles Sehen und Darstellen, Technische Universität Graz. www.icg.tu-graz.ac.at, 2001.

[Log01] Logitech. www.logitech.com, 2001.

[MC95] J. Martin and J. Crowley. Comparison of correlation techniques, 1995.

[Mic01] Microsoft. MFC Fundamentals. www.msdn.microsoft.com, 2001.

[RC78] T.W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *SMC*, 8(8):629–632, August 1978.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[SHB93] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing and Machine Vision.* Chapman & Hall Computing, 1993.

[Soi99] P. Soille. *Morphological Image Analysis.* Springer-Verlag, Berlin, 1999.

[SZ98] S. Scherer and A. Zaworka. Machbarkeitsstudie "Golden Eye". Technical report, Institute for Computer Graphics and Vision, Graz University of Technology, 1998.

[TWA01] TWAIN Working Group. www.twain.org, 2001.

[Vid01] Video for Windows. www.videoforwindows.com, 2001.

[ZS00] A. Zaworka and S. Scherer. Machine Vision Driven Real-time Black Jack Analysis. In *Proceedings of 24rd Workshop of the AAPR.* Austrian Association for Pattern Recognition, 2000.